3 cad evolution

Manuale Tecnico di Progettazione

DAU
Tecnologie applicate al software

info@3cadevolution.it

Versione attuale: 1.3.2
19 marzo 2013

Sommario

presente documento contiene il manuale tecnico di progettazione di ***3CadEvolution*** .

Informazioni documento

Titolo:	Manuale Tecnico di Progettazione
Nome file:	ManualeTecnico-1.3.2.pdf
Data creazione:	11/06/2012
Data ultima modifica:	11/03/2013
Versione attuale:	1.3.2
Distribuito da:	Dau S.r.l.

copyright ©

Tutti i diritti riservati. Il marchio ***3CadEvolution*** è registrato e di proprietà di Consorzio 3CAD.

3CAD è un consorzio che nasce dall'esperienza pluriennale di tre società (APRA SISTEMI, DAU e In.Tres) specializzate in soluzioni software di tipo tecnico-gestionale per il settore del mobile. La costituzione di un unico gruppo ha permesso l'integrazione delle diverse applicazioni informatiche utilizzate nelle varie fasi della filiera del mobile e lo sviluppo di un esclusivo sistema che riunisce la progettazione tecnica, la gestione della produzione e la vendita. Perciò 3CAD è in grado di fornire una offerta globale nella progettazione, implementazione e manutenzione di soluzioni CAD/CAM/CIM per l'industria del mobile e può garantire assistenza tecnica e commerciale sul territorio italiano.

Indice

1 Introduzione **1**
- 1.1 Definizione dell'utente del prodotto . . . 1
- 1.2 Come leggere il manuale . . . 1

2 Definizione del prodotto **2**
- 2.1 Terminologia del prodotto . . . 3
- 2.2 Requisiti . . . 4
- 2.3 Installazione del programma . . . 5
- 2.4 Installazione di un catalogo . . . 13
- 2.5 ***eCadUtil*** - Parametri da linea di comando . . . 15
- 2.6 Database . . . 16
- 2.7 Struttura dell'applicazione e caratteristiche . . . 18
  - 2.7.1 Organizzazione dei file . . . 18
  - 2.7.2 Cartella procedure . . . 20
  - 2.7.3 File di configurazione . . . 22
  - 2.7.4 Numerazione degli ordini . . . 22
  - 2.7.5 Salvataggio dell'ordine (Flag) . . . 24
  - 2.7.6 Impostazione di aggiornamento automatico del programma . . . 25

3 Creazione di un nuovo catalogo **26**
- 3.1 Partenza . . . 26
- 3.2 Creazione di un mobile componibile . . . 28
- 3.3 Variabili . . . 31
- 3.4 Varianti . . . 32
- 3.5 Variabili di sistema . . . 38
- 3.6 Condizioni . . . 39
- 3.7 Tipologie e default . . . 40
- 3.8 Dragger . . . 45
- 3.9 Creazione di una macro e riutilizzo del codice . . . 48
- 3.10 Creazione della variante di tipo frontale . . . 50
  - 3.10.1 Impostazione e modifica di alcuni materiali . . . 51
  - 3.10.2 Modifica della variante *colore* . . . 52
  - 3.10.3 Impostazione delle altre varianti . . . 56
  - 3.10.4 Impostazione interferenza . . . 60
  - 3.10.5 Tipologie a riconduzione . . . 66
  - 3.10.6 Raggruppamento delle variabili in un gruppo . . . 68
  - 3.10.7 Oggetti 3D esterni . . . 73
- 3.11 Regole con le aggiunte . . . 74
- 3.12 Regole con le divisioni . . . 78
- 3.13 Regole con il placer . . . 82
- 3.14 Placer multicatalogo . . . 83
- 3.15 Macro di più articoli . . . 84
- 3.16 Distinta base a formule e listino a formule . . . 86
- 3.17 Distinta base estesa . . . 89
  - 3.17.1 Installazione distinta base estesa . . . 90
  - 3.17.2 Gestione derive varianti . . . 91

4 Progettazione su regole **92**
- 4.1 Definizione della regola . . . 92

4.2 Varianti della regola . . . 97
4.3 Gruppi di valutazione . . . 100
4.4 Sintassi delle regole . . . 100
4.4.1 Caratteri riservati . . . 102
4.4.2 La calcolatrice di ***3CadEvolution*** . . . 107
4.4.3 Operatori logici . . . 109
4.4.4 Confronto di stringhe . . . 110
4.4.5 Operatori numerici . . . 110
4.5 Variabili di sistema di utilità . . . 112
4.6 Comandi del linguaggio delle regole . . . 114
4.6.1 Comando *#LET* . . . 114
4.6.2 Comando *#LETS* . . . 115
4.6.3 Comando *#LETV* . . . 116
4.6.4 Comando *#LETS_MACROFINALE* . . . 116
4.6.5 Comandi *#IF #ELSEIF #ENDIF* . . . 117
4.6.6 Comando *#IF* . . . 119
4.6.7 Comandi *#IFDEF #IFNDEF* . . . 119
4.6.8 Comando *#FOR* . . . 121
4.6.9 Ciclo *#DO...#LOOP (#continue) (#break)* . . . 123
4.6.10 Comando *#BREAK* . . . 123
4.6.11 Comando *#CONTINUE* . . . 123
4.6.12 Comando *#SQL* . . . 124
4.6.13 Comando *#SELECT* . . . 124
4.6.14 Comando *#SWAP* . . . 125
4.6.15 Comando *#MSG* . . . 125
4.6.16 Comandi *#PUSH* e *#POP*, e *#POPALL* . . . 125
4.6.17 Comando *#DUMP* . . . 126
4.6.18 Comando *#INP* . . . 126
4.6.19 Comandi *#GOTO* o *#JUMP* o *#VAI* . . . 126
4.6.20 Script *#VB* e *#ENDVB* . . . 127
4.6.21 Comandi *#FORDBS* e *#ENDDBS* . . . 128
4.6.22 Comando *#FORFILE* . . . 128
4.7 Comandi estesi per la grafica . . . 129
4.7.1 Comandi *!QUOTAxxx* e *!QUOTE* . . . 129
4.7.2 Comando *!PM* . . . 132
4.7.3 Comando *!ALTEZZE* . . . 133
4.7.4 Comandi *!MTTRASLA* e *!MTINIT* o *!MTRESET* . . . 133
4.7.5 Comandi *!ADDPUNTO*, *!DELPUNTO* e *!AZZERAPUNTI* . . . 134
4.7.6 Comandi *!INVERITLATO* e *!AZZERAINVERTILATO* 134
4.7.7 Comando *!COMMNETO* . . . 134
4.7.8 Comando !FLAGSPEC . . . 135
4.7.9 Comandi per le informazioni sull'articolo . . . 135
4.7.10 Comandi *!ADDLINK*, *!SCEGLIADDLINK*, *!TIPOBOX* 135
4.7.11 Comando *!ADDMAGIC* . . . 140
4.7.12 Comando *!IMGSCALA* . . . 141
4.7.13 Comando *!LASER* . . . 141
4.7.14 Comandi *!CODICEBARRA*, *!DIMBARRA* . . . 142
4.8 Comandi per la gestione delle sagome . . . 143
4.8.1 Comando *!SAGOMA* . . . 143
4.8.2 Comando *!SAGDUMP* . . . 147

4.8.3 Comandi *!SAGMID, !SAGPERCX, !SAGPERCY* . . . 147
4.8.4 Comando *!SAGXY* . . . 148
4.8.5 Comando *!SAGOFFSET* . . . 149
4.8.6 Comandi *!SAGCLIPX, !SAGCLIPY* . . . 149
4.8.7 Comandi *!SAGCLIPR, !SAGCLIPL, !SAGCLIPU, !SAGCLIPD* . . . 150
4.8.8 Comando *!FRESA* . . . 153
4.8.9 Comando *!LAMIERA* . . . 158
4.8.10 Comando *!SAGMIRROR* . . . 158
4.8.11 Comandi *!SAGUNION, !SAGDIFF* . . . 160
4.8.12 Comando *!SAGREL* o *!SAGSPOSTA* . . . 162
4.8.13 Esempio complesso della gestione delle sagome . . . 164
4.9 Comandi matematici delle regole . . . 173
4.9.1 Comando *!MTCOLLEGA* . . . 173
4.9.2 Comando *!DISTANZA* . . . 173
4.9.3 Comando *!RETTA* . . . 173
4.9.4 Comando *!RETTA2* . . . 174
4.9.5 Comando *!MATHAZZERA* . . . 174
4.9.6 Comando *!INTERSECA* . . . 175
4.9.7 Comando *!PUNTO* . . . 175
4.9.8 Comando *!PUNTOL* . . . 175
4.9.9 Comando *!PUNTOREL* . . . 175
4.10 Comandi per il disegno 2D . . . 176
4.10.1 Comando *!!RECT* . . . 176
4.10.2 Comando *!!LINE* . . . 176
4.10.3 Comando *!!POLY* . . . 176
4.10.4 Comando *!!CIRCLE* . . . 177
4.10.5 Comando *!!ARC* . . . 177
4.10.6 Comando *!!PIE* . . . 177
4.10.7 Comando *!!FILLRECT* . . . 177
4.10.8 Comando *!!FILLCIRCLE* . . . 178
4.10.9 Comando *!!FILLPOLY* . . . 178
4.10.10 Comando *!!STRING* . . . 178
4.10.11 Comando *!!PUSHFONT* . . . 178
4.10.12 Comando *!!POPFONT* . . . 178
4.10.13 Comando *!!PUSHMATRIX* . . . 179
4.10.14 Comando *!!POPMATRIX* . . . 179
4.10.15 Comando *!!TRASLA* . . . 179
4.10.16 Comando *!!RUOTA ANGOLO* . . . 179
4.10.17 Comando *!!LINETO* . . . 179
4.10.18 Comando *!!MOVETO* . . . 180
4.10.19 Comando *!!DXF* . . . 180
4.10.20 Comando *!!DXF2* . . . 180
4.10.21 Comando *!!BRECT* . . . 180
4.11 Funzioni . . . 181
4.11.1 Funzione `LOOK` . . . 181
4.11.2 Funzione `LOOKB` . . . 182
4.11.3 Funzione `IIF` . . . 182
4.11.4 Funzione `CASE` . . . 182
4.11.5 Funzione `DIR` . . . 183
4.11.6 Funzione `LEN` . . . 183

4.11.7 Funzione MID . . . 184
4.11.8 Utilizzo funzione ESTRAI o EX . . . 184
4.11.9 Funzione INSTR . . . 185
4.11.10 Funzione FORMAT . . . 185
4.11.11 Funzione FORMATV . . . 186
4.11.12 Funzione SQL e SQLA . . . 186
4.11.13 Funzione CODVAR . . . 186
4.11.14 Funzione VAR o VAROPZ . . . 187
4.11.15 Funzione VARIANTE . . . 187
4.11.16 Funzione WEEK . . . 187
4.11.17 Funzione DATA . . . 188
4.11.18 Funzione DIM . . . 188
4.11.19 Funzione LIRE, funzione EURO . . . 188
4.12 Variabili di sistema per la grafica . . . 189
4.13 Variabili . . . 192
4.14 Attivazione delle tipologie . . . 193
4.14.1 Parametri di attivazione delle tipologie . . . 193
4.14.2 Utilizzo e gestione delle macro di tipologie . . . 195
4.14.3 Macro Top . . . 196

5 Stampe parametriche 198
5.1 Gruppi della regola di stampa . . . 198
5.2 Variabili della regola di stampa . . . 199
5.3 Istruzioni per la formattazione della pagina . . . 202
5.4 Parametri per i comandi *ARG* e *TOP* . . . 213
5.4.1 Parametro F . . . 215
5.4.2 Parametro TI . . . 215
5.4.3 Parametro TE . . . 215
5.4.4 Parametro PA . . . 215
5.4.5 Parametro BI . . . 216
5.4.6 Parametro BR . . . 216
5.4.7 Parametro CO . . . 216
5.4.8 Parametro TX . . . 216
5.4.9 Parametro LS . . . 216
5.4.10 Parametro BXI . . . 217
5.4.11 Parametro BXE . . . 217
5.4.12 Parametro BXT . . . 217
5.4.13 Parametro SL . . . 217
5.4.14 Parametro BW . . . 217
5.4.15 Parametro EX . . . 217
5.4.16 Parametro Z . . . 217
5.4.17 Parametro SCRIPT . . . 218
5.4.18 Parametro GA . . . 218
5.4.19 Parametro CA . . . 218
5.4.20 Parametro BXTOPFRONT . . . 219
5.5 Parametri del comando *ARG* . . . 219
5.5.1 Parametro ORDINA . . . 219
5.5.2 Parametro PR . . . 220
5.5.3 Parametro AS . . . 220
5.5.4 Parametro BOX . . . 220
5.5.5 Parametro SCALA . . . 220

5.5.6 Parametro QUOTE . . . 221
5.5.7 Parametro VC . . . 221
5.5.8 Parametro ANGLE . . . 221
5.5.9 Parametro ETI . . . 222
5.6 Parametri del comando *TOP* . . . 222
5.6.1 Parametro QC . . . 222
5.6.2 Parametro QD . . . 222
5.6.3 Parametro M . . . 222
5.6.4 Parametro T . . . 223
5.7 Comando *NTAB* . . . 225
5.8 Stampa per parete . . . 226
5.9 Stampa con righelli(Rullers) . . . 229
5.10 Stampa con marker . . . 230
5.11 Stampa dei listini . . . 234

6 Lavorazioni 235
6.1 Premesse . . . 237
6.2 Interfaccia . . . 238
6.3 Variabili . . . 241
6.3.1 Variabili della tipologia corrente . . . 241
6.3.2 Variabili della tipologia appoggiata . . . 241
6.4 Creazione della lavorazione . . . 242
6.5 Elaborazioni . . . 244
6.6 File . . . 248
6.6.1 File *Distinta.CFG* . . . 248
6.6.2 File *Disegno.CFG* . . . 248
6.6.3 File *Dis3d.CFG*. . . . 250
6.7 Postprocessor . . . 251
6.8 Lavorazioni tra componenti a riconduzione . . . 252

7 Gestione degli imballi 254
7.1 Interfaccia . . . 254
7.1.1 Flag colli . . . 257
7.1.2 Altri dati . . . 259
7.2 Imballi associati . . . 260
7.3 Dichiarazione in anagrafica articolo . . . 262
7.4 Funzione `dau_getimballo` . . . 263

8 Costruzione delle finiture 265
8.1 Funzioni per la definizione delle finiture . . . 266
8.1.1 Funzione `wrFaccia` . . . 266
8.1.2 Funzione `wrFacciaAlfa` . . . 266
8.1.3 Funzione `wrScala` . . . 267
8.1.4 Funzione `wrParametro` . . . 267
8.1.5 Funzione `wrTexAntaCurva` . . . 267
8.1.6 Funzione `wrPanExt` . . . 268
8.1.7 Funzione `wrPannello` . . . 268
8.1.8 Funzione `wrPannello2` . . . 269
8.1.9 Funzione `wrDecalco` . . . 269
8.1.10 Funzione `wrTrasforma` . . . 270
8.1.11 Funzione `wrTrasformaPannello` . . . 270
8.1.12 Funzione `wrChiudi` . . . 271

8.1.13 Funzione wrSfera 271
8.1.14 Funzione wrBox 271
8.1.15 Funzione wr3D 272
8.1.16 Funzione wr3D2 272
8.1.17 Funzione wEstrudi 273
8.1.18 Funzione wrFacciaAlfa 273
8.1.19 Funzione wrTexAnta 275
8.1.20 Funzione wrSpeciale 275
8.1.21 Funzione wrCilindro 276
8.1.22 Funzione WrToro 276
8.1.23 Funzione wColore 277
8.1.24 Funzione wrLibreria 278
8.1.25 Funzione wrLinea 278
8.1.26 Funzione wrLinee 278
8.1.27 Funzione wrMuro 279
8.1.28 Funzione wrRivoluzione 279
8.1.29 Funzione wrIniSagoma 280
8.1.30 Funzione wrYSagoma 280
8.1.31 Funzione wrSimbolo 280
8.1.32 Funzione wrText 281
8.1.33 Funzione wrXMov 281
8.1.34 Lista completa delle funzioni disponibili 282
8.2 Utilizzo file .x 286
8.2.1 Funzioni WRX e WRX2 287

9 Funzione elabora finale **291**
9.1 Public Function JobMain 292
9.2 Public sub JobIni 294
9.3 Public Sub JobPannello 295
9.4 Public sub JobFin 297

10 Print Server **298**
10.1 Configurazione 298
10.2 Stampa ordini 299
10.3 Download ordini 301
10.4 EvoStarter 303
10.5 Elenco chiamate script utilizzate dal Print Server 303

A Glossario **305**

Elenco delle figure

2.1 Loader dell'installazione di ***3CadEvolution*** 5
2.2 Lancio dell'utility dxDIAG. 6
2.3 Informazioni su componenti e driver ***DirectX*** installati nel sistema. 7
2.4 Selezione percorso dati. 8
2.5 Universal data link. 9
2.6 Installazione ***SQL Server*** Express 2008 R2. 9
2.7 Selezione lingua. 10
2.8 Inizio setup. 10
2.9 Selezione della cartella di installazione. 11
2.10 Selezione della cartella nel menu Avvio/Start. 11
2.11 Creazione icone Desktop/Avvio veloce. 12
2.12 Avvio installazione. 12
2.13 Installazione catalogo. 13
2.14 Manutenzione database. 16
2.15 Area errori e log. 18
2.16 Cartella `_ECADPRO`. 19
2.17 Schermata di ecadpro.ini 22
3.1 Menu procedure. 26
3.2 Inserimento nome catalogo. 27
3.3 Articolo di test. 28
3.4 Modifica articolo. 29
3.5 Dettaglio articolo. 29
3.6 Schema di modifica della regola(Help Ogetto). 29
3.7 Editor regola. 32
3.8 Variante che imposta gli spessori. 34
3.9 Editor varianti. 34
3.10 Inserimento nuova colonna. 35
3.11 Variante spessore. 35
3.12 Visualizzazione a video del contenuto di una variabile 36
3.13 Creazione variante *MISURA*. 36
3.14 Collegamento tra la variante *TIPO* e la variante *MISURA*. . . 37
3.15 Form di variazione altezza, larghezza e profondità. 39
3.16 Zoccolo dx e sx. 40
3.17 Interfaccia di gestione della tipologia. 41
3.18 Tipologia 002 rappresentante Fianco. 43
3.19 Sottotipologia 002.001 rappresentante Fianco DX. 43
3.20 Tipologia 003 rappresentante Ripiano. 43
3.21 Tipologia 005 rappresentante Schienale. 43
3.22 Esportazione dati. 44
3.23 Mobile con dimensioni variabili. 46
3.24 Mobile con dimensioni variabili e utilizzo della variante *RADDOPPIA*. 46
3.25 Variante raddoppia. 49
3.26 Definizione di un materiale. 51
3.27 Materiale legno. 51
3.28 Modifica variante colore. 53
3.29 Materiali (1). 53
3.30 Impostazione dei livelli per la variante colore. 54

3.31 Materiali (2). 54
3.32 Mobile con il materiale legno applicato. 55
3.33 Modifica variante Tipo Finitura. 56
3.34 Menu setup, importazione e esportazione dei dati. 58
3.35 Lista delle tipologie. 58
3.36 Posizione maniglia. 59
3.37 Parametri aggiuntivi. 59
3.38 Impostazione interferenza. 61
3.39 Variante di tipo *separatore*. 62
3.40 Sottotipologia 004.001, ripiano vetro. 62
3.41 Tipologia a riconduzione 020, gestione del top. 66
3.42 Paramentri speciali. 67
3.43 Varianti prima del raggruppamento. 68
3.44 Editor gruppi, variante FINITURA. 68
3.45 Variante FINITURA di tipo gruppo. 68
3.46 Editor gruppi, variante STRUTTURA. 69
3.47 Variante STRUTTURA di tipo gruppo. 69
3.48 Editor regola per un box sagomato. 69
3.49 Associare regola ad un articolo. 70
3.50 Disegno in CAD2D. 70
3.51 Mobile sagomato. 72
3.52 Maniglia con utilizzo del 3DS esterno. 74
3.53 Mobile con tre vani. 78
3.54 Regola con le divisioni, variante DIVVANI. 79
3.55 Editor gruppi, DIVVANI. 79
3.56 Parametri divisore. 79
3.57 Oggetto devider. 82
3.58 Mobile con utilizzo dei divider. 82
3.59 Il simbolo in grafica del placer multicatalogo. 84
3.60 Macro di più articoli. 85
3.61 Gestione delle tabelle, tabella RP. 87
3.62 Inserimento in anagrafica articoli dei codici. 87
3.63 Modifica articolo, SFIA. 87
3.64 Disegno del Box, articolo 001. 88
3.65 Dettaglio della regola rappresentante il Box. 88
4.1 Modifica regola, pannello principale. 92
4.2 Editor regola. 93
4.3 Flag, variabili di completamento. 95
4.4 Campi di variabilità delle dimensioni. 95
4.5 Causali fuori misura. 97
4.6 Posizione anta. 133
4.7 Rotazione box. 140
4.8 Collegamento box su linee base 1. 141
4.9 Collegamento box su linee base 2. 141
4.10 Editor tipologia, associazione della sagoma. 144
4.11 Sagoma semplice. 144
4.12 Sagoma a forma di ellisse. 145
4.13 Sagoma a forma di ellisse. 146
4.14 Doppia estrusione della sagoma secondo l'altezza. 146
4.15 Output prodotto dal comando !SAGDUMP. 147
4.16 Output comando !SAGPERCX. 148

4.17 Output comando !SAGXY. 148
4.18 Taglio della sagoma lungo l'asse X. 149
4.19 Taglio della sagoma lungo l'asse Y. 150
4.20 Controsagoma della parte destra della sagoma. 150
4.21 Controsagoma dell'intera sagoma distanziata 50mm. 152
4.22 Direzioni x e y dei lati per la fresa. 154
4.23 Sagoma con la fresa circolare semplice. 155
4.24 Due frese sovrapposte. 156
4.25 Fresa con utilizzo della rotazione. 157
4.26 La stessa fresa su quattro lati diversi. 157
4.27 Lamiera con spessore 10. 158
4.28 Lamiera con spessore 50. 158
4.29 Sagoma di partenza, utilizzo comando !SAGMIRROR. 159
4.30 Sagoma specchiata nell'asse X. 159
4.31 Sagoma specchiata nell'asse Y. 159
4.32 Sagoma specchiata nell'asse XY. 159
4.33 Unione di due sagome. 160
4.34 Unione di due sagome circolari. 161
4.35 Differenza di due sagome. 161
4.36 Differenza di due sagome circolari. 161
4.37 Un pannello orizzontale ed uno verticale con applicazione del materiale legno non corretta. 163
4.38 Un pannello orizzontale ed uno verticale con applicazione del materiale legno corretta. 164
4.39 Mobile sagomato. 165
4.40 Gli archi della sagoma. 165
4.41 I fianchi sagomati. 166
4.42 Applicazione del comando !SAGMID. 167
4.43 Ripiano sagomato. 168
4.44 Due ripiani sagomati. 169
4.45 Necessità dell'utilizzo del comando !SAGCLIPY. 170
4.46 Inserimento della base, cappello e schienale del mobile. 170
4.47 Top sporgente. 171
4.48 Anta sagomata. 172
4.49 Retta (1). 174
4.50 Retta (2). 174
5.1 Regola di stampa. 198
5.2 Il 20% del parametro gamma. 218
5.3 Il 70% del parametro gamma. 218
5.4 Il 100% del parametro gamma. 218
5.5 Immagini rappresentanti il chiaro/scuro delle stampe. 218
5.6 Stampe con ruller. 231
5.7 Posizione pallino marker. 232
6.1 Schema funzionamento job. 236
6.2 Orientamento delle tipologie per il job. 237
6.3 Orientamento delle tipologie per il job. 238
6.4 Orientamento delle tipologie. 238
6.5 Interfaccia della tipologia. 239
6.6 Ambienti per le tipologie di lavorazione. 239
6.7 Dichiarazione delle tipologie a contatto. 240
6.8 Dichiarazione del lato sensibile per la lavorazione. 240

6.9 Schema delle variabili della tipologia appogiata. 242
6.10 Box elementare. 242
6.11 Lavorazioni per contatto, coperchio e fondo. 243
6.12 Editor per la regola di foratura. 243
6.13 Schema di spiegazione di disposizione delle facce per le lavorazioni. 244
6.14 Elaborazioni. 245
6.15 Disegno della lavorazione. 245
6.16 Elaborazioni schermata generale. 246
6.17 Schermata del disegno della lavorazione. 247
6.18 Articolo con ferramenta durante l'esplosione. 251
6.19 Boiserie accostate e una pedana al metro lineare, mobile che necessita la lavorazione a riconduzione. 253
7.1 Interfaccia imballi. 254
7.2 Interfaccia imballi, intestazione. 254
7.3 Interfaccia imballi, caratteristiche. 255
7.4 Calcolo delle dimensioni e del volume nel imballo. 256
7.5 Calcolo delle dimensioni e del volume nel imballo. 256
7.6 Tabella per il calcolo delle dimensioni. 257
7.7 Colli. 257
7.8 Ordinamento dei colli. 259
7.9 Altri dati per imballi. 260
7.10 Imballi associati. 260
7.11 Imballi associati, compilazione. 261
7.12 Distinta base collo. 261
7.13 Anagrafica articoli, dichiarazione imballo. 262
7.14 Funzione `dau_getimballo`. 263
8.1 Editor help per inserire lo script delle ante. 265
8.2 Estrusione elica. 274
8.3 Esterno. 289
8.4 Interno. 289
10.1 Impostazione Print Server. 299
10.2 Visualizzazione degli ordini da stampare. 300
10.3 Visualizzazione download ordini. 302

Elenco delle tabelle

3.1 Variabili di sistema. 38
3.2 Spostamento box. 47
4.1 Tipo di posizionamento box nella grafica. 96
4.2 Tipi di varianti. 100
4.3 Caratteri riservati. 103
4.4 Funzioni numeriche della calcolatrice. 109
4.5 Variabili di sistema. 114
4.6 Link utilizzati dal comando *!ADDLINK*. 137
4.7 Numeri per i collocamenti all'interno del gruppo. 139
4.8 Variabili di sistema per la grafica. 189
4.9 Variabili di sistema per la grafica legate ai pulsanti. 190
4.10 Variabili di sistema per la grafica legate ai box. 192
4.11 Parametri per le tipologie. 195
5.1 Variabili predefinite standard per le stampe. 201
5.2 Variabili predefinite gestione carichi per le stampe. 201
5.3 Variabili di programmazione per le stampe. 202
5.4 Tipo di barcode. 210
5.5 Tipi di quotature per la stampa pareti. 228
6.1 Colori per il file *Disegno.CFG* 249
6.2 Stile di linea per il file *Disegno.CFG* 250
6.3 Oggetti da disegnare per il file *Disegno.CFG* 250
8.2 Tipi di strecciabilità. 289
9.1 Variabili per job finale. 293
9.2 Variabili della funzione JobPannello. 296

Elenco degli esempi

2. 1 Utilizzo del parametro /AUTO di ***eCadUtil*** 15
2. 2 Utilizzo del parametro /OFFLINE di ***eCadUtil*** 15
2. 3 Utilizzo del parametro /SWAP di ***eCadUtil*** 16
2. 4 File *aggiornamenti.ini* 25
3. 1 Cambio Dim.L attraverso una regola. 30
3. 2 Due pannelli di spessore 30mm. 30
3. 3 Assegnazione alla Dim.L del valore di una variabile attraverso una regola. 31
3. 4 Assegnazione di un calcolo algebrico come valore di una variabile. 31
3. 5 Utilizzo delle variabili all'interno della regola. 32
3. 6 Variante spessore rappresentata nella grafica. 34
3. 7 Messaggio a video, visualizzazione del valore di una variabile. . . 36
3. 8 Passaggio del valore di una variabile come parametro di un'altra (1). 38
3. 9 Passaggio del valore di una variabile come parametro di un'altra (2). 38
3. 10 Spostamento del punto d'origine di un box. 39
3. 11 Colore espresso in RGB 42
3. 12 Script di impostazione delle tipologie. 44
3. 13 Regola con utilizzo del della dragger. 46
3. 14 Creazione della macro BOX. 48
3. 15 Utilizzo della macro BOX. 48
3. 16 Utilizzo della variante raddoppia all'interno della regola. 49
3. 17 Regola rappresentante un box doppio. 50
3. 18 Macro di assegnazione che imposta le variabili principali. 50
3. 19 Collegamento della variante colore alla visualizzazione. 55
3. 20 Posizionamento degli oggetti. 61
3. 21 Macro BOX complessa. 62
3. 22 Condizione del tipo di materiale per i ripiani. 63
3. 23 Condizione del comando for. 64
3. 24 Tipo di ripiano. 64
3. 25 Box complesso. 65
3. 26 Macro cassetti. 65
3. 27 Variante *vTop* di tipo sì/no. 67
3. 28 Utilizzo sagoma. 71
3. 29 Regola completa che definisce 3 sagome. 71
3. 30 Integrazione del nome di un 3DS. 73
3. 31 Regola del box che si ridimensiona automaticamente per ogni aggiunta. 75
3. 32 Forzare la rivalutazione dei figli per i placer. 83
3. 33 Forzare la rivalutazione del padre. 83
3. 34 Definizione di un placer multicatalogo. 84
4. 1 Utilizzo caratteri //(1) 103
4. 2 Utilizzo caratteri //(2) 104
4. 3 Utilizzo carattere \. 104
4. 4 Utilizzo carattere *. 104
4. 5 Utilizzo caratteri { }. 105
4. 6 Comando $ per generare una stringa di numeri. 106
4. 7 Assegnazione di un valore numerico ad una variabile. 109

4. 8 Assegnazione di un valore restituito da una funzione numerica. . 109
4. 9 Utilizzo operatore logico &. 109
4. 10 Confronto tra due stringhe, se la prima è minore della seconda. . 110
4. 11 Confronto tra due stringhe, se sono uguali. 110
4. 12 Divisione. 110
4. 13 Generare un numero random. 112
4. 14 Assegnare un valore alla variabile. 114
4. 15 Assegnare ad una variabile una stringa. 115
4. 16 Assegnazioni multiple. 116
4. 17 Regola dove si utilizza il divider 117
4. 18 *#IF* annidate. 118
4. 19 *#IF* struttura completa. 119
4. 20 *#IF* senza il relativo *#ENDIF*. 119
4. 21 Cicli FOR annidati. 122
4. 22 Utilizzo comando *#CONTINUE*. 123
4. 23 Costrutto *#SELECT*. 125
4. 24 Utilizzo comando *#GOTO*. 126
4. 25 Inserimento degli script all'interno di una regola. 127
4. 26 Utilizzo comando *#FORDBS*. 128
4. 27 Utilizzo comando *#FORFILE*. 129
4. 28 Comando *!QUOTA*. 129
4. 29 Due quote differenti. 130
4. 30 Creazione delle quotature senza il comando *!QUOTA*. 130
4. 31 Utilizzo comando *!QUOTE*. 131
4. 32 Quota con la sintassi alternativa per i punti. 132
4. 33 Utilizzo comando *!COMMNETO*. 135
4. 34 Posizionamento box a 90 gradi sullo spigolo destro. 140
4. 35 Utilizzo comando *!ADDMAGIC* 140
4. 36 Utilizzo comando *!IMGSCALA*. 141
4. 37 Utilizzo comandi *!CODICEBARRA*, *!DIMBARRA* 142
4. 38 Sagoma con arco(1). 144
4. 39 Sagoma con arco(2). 145
4. 40 Sagoma a forma di ellisse. 145
4. 41 Sagoma a forma di cerchio. 145
4. 42 Utilizzo comando *!SagomaC*. 146
4. 43 Utilizzo comando *!SAGOFFSET*. 149
4. 44 Comando *!FRESA*. 153
4. 45 Frese sovrapposte. 155
4. 46 Comando *!FRESA*. 155
4. 47 Comando *!FRESA*. 156
4. 48 Utilizzo comando *!SAGMIRROR*. 160
4. 49 Utilizzo comando *!SAGSPOSTA*. 162
4. 50 Utilizzo comando *!!DXF* 180
4. 51 utilizzo funzione `LOOK` 181
4. 52 Utilizzo funzione `LOOKB`. 182
4. 53 Utilizzo funzione `IIF`. 182
4. 54 Utilizzo funzione `CASE` 183
4. 55 Utilizzo funzione `DIR` 183
4. 56 Utilizzo funzione `LEN` 183
4. 57 Utilizzo funzione `MID` completa 184
4. 58 Utilizzo funzione `MID` incompleta 184

4. 59 Utilizzo funzione ESTRAI. 184
4. 60 Utilizzo funzione INSTR 185
4. 61 Utilizzo funzione FORMAT (1). 185
4. 62 Utilizzo funzione FORMAT (2) 185
4. 63 Utilizzo funzione FORMATV 186
4. 64 Utilizzo funzione SQL 186
4. 65 Utilizzo funzione CODVAR 187
4. 66 Utilizzo funzione VARIANTE 187
4. 67 Utilizzo unzione WEEK 187
4. 68 Utilizzo funzione DATA. 188
4. 69 Utilizzo funzione DIM 188
4. 70 Utilizzo funzione EURO 188
4. 71 Scrittura di una regola 190
4. 72 Attivazione di tipologia semplice e di tipologia con parametri. . 193
4. 73 Parametro SE 195
4. 74 Macro CASSETTO 195
4. 75 Regola che disegna una cassettiera con quattro cassetti. 196
4. 76 Costruzione del codice in funzione della profondità del piano . . 196
5. 1 Utilizzo dei comandi DIM e FONT nella stampa. 203
5. 2 Utilizzo comando LI 204
5. 3 Utilizzo comando TX 205
5. 4 Utilizzo comando FONT 206
5. 5 Utilizzo comando WMF 206
5. 6 Utilizzo comando CODE25 207
5. 7 Utilizzo comando CODE39. 208
5. 8 Utilizzo comando COLORPENNA 211
5. 9 Utilizzo comando COLORERIEMPIMENTO 212
5. 10 Utilizzo comando TIPORIEMPIMENTO 212
5. 11 Utilizzo comando SPESSORE 213
5. 12 Utilizzo comando ORIGINE 213
5. 13 Utilizzo istruzione ARG. 214
5. 14 Stampa del disegno in vista ortogonale escludendo alcune tipologie. 215
5. 15 Utilizzo parametro del /GA di ARG. 218
5. 16 Regola di stampa con /BXTOPFRONT. 219
5. 17 Stampa delle quote globali. 221
5. 18 Utilizzo istruzione TOP. 223
5. 19 Stampa di una tabella con /NTBA 225
5. 20 Stampa. 226
5. 21 Stampa per parete. 227
5. 22 Layer di quotatura per la stampa pareti (1). 228
5. 23 Layer di quotatura per la stampa pareti (2). 228
5. 24 Marker nella stampa. 232
6. 1 Estrapolazione del parametro della tipologia dalla grafica. 241
7. 1 Ordinamento dei componenti in distinta base. 257
7. 2 Dichiarazione di un componente come montato. 258
8. 1 Inserimento anta all'interno della regola. 265
8. 2 Utilizzo della funzione wrPannello2. 269
8. 3 Estrusione elica con il linguaggio delle regole. 274
8. 4 Creazione di una *Bump map*. 277
8. 5 Utilizzo file .x 290
9. 1 Flag elaborafinale. 291

9. 2 Script per modificare alcune varianti nella regola. 293
9. 3 Utilizzo funzione JobIni. 294
9. 4 Utilizzo funzione JobPannello. 297
10. 1 Impostazione di tutte le funzionalità di Print Server contemporaneamente. 299
10. 2 Creazione del disegno ordine in formato JPG. 301

Elenco dei codici

1 Assegnazione di un valore alla variabile L 30
2 La stessa tipologia con parametri diversi. 30
3 Variabile spessore. 31
4 Calcolo algebrico come valore della variabile con *#LET*. 31
5 Variabili all'interno di un calcolo algebrico assegnato come valore di una variabile. 32
6 Variante spessore parametrizzata. 34
7 Assegnazione della variabile spessore ai fianchi. 35
8 Comando *#MSG*, messaggio a video. 36
9 Variabile tipo di spessore. 37
10 Variabile misura. 37
11 Cambiamento del codice della regola in base ai valori inseriti all'interno della tabella di inserimento delle variabili. 37
12 Variabile colonna passata come parametro alla variabile tipo. . 38
13 Nome della variabile con un parametro (indice). 38
14 L, sl dipendenti dalle variabili. 39
15 Tipologie esportate. 44
16 Regola per il mobile. 45
17 Definizione di un dragger. 46
18 Macro BOX. 48
19 Contenuto della macro BOX. 48
20 Utilizzo della macro BOX. 48
21 Variante raddoppia. 49
22 Modifiche alla macro BOX necessarie all'utilizzo della variante spessore. 49
23 Box doppio. 50
24 Macro ASSEGNA. 50
25 Schema delle caratteristiche dei materiali. 52
26 Macro ASSEGNA con la definizione del colore base. 55
27 Utilizzo di interferenza. 61
28 Macro BOX. 63
29 Variante *TipoF*. 64
30 Macro ante. 64
31 Integrazione delle ante. 65
32 Macro _Cassetti. 65
33 Gruppo della macro cassetti all'interno del box. 66
34 Modifiche al BOX con l'aggiunta della variante *vTop*. 67
35 Regola complessa. 70
36 Inserimento sagoma. 71
37 Regola completa di un mobile con utilizzo delle sagome. 71
38 Tipologia di test. 73
39 Inserimento di un 3DS. 73
40 Box con le aggiunte. 75
41 Logica di funzionamento delle aggiunte. 75
42 VanoAggiunta. 76
43 VANOFINE. 77
44 DivStruttura e DivVani. 80
45 Macro DivStruttura. 81
46 Macro DivVani. 81

47 Parametro _PLACERRIVFIGLI. 83
48 Parametro _PLACERRIVPADRE. 83
49 Placer multicatalogo. 84
50 Regola con le macro di più articoli. 85
51 Regola per anagrafica articoli. 88
52 Gruppo [INFO]. 101
53 Gruppo [GENE]. 101
54 Gruppi [PREV], [CODICE], [GRAFICA]. 102
55 Inserimento di un commento. 103
56 Andare a capo. 104
57 Troncamento della riga in caso la variabile non sia definita. . . 104
58 Conversione di un calcolo algebrico in una stringa. 105
59 Carattere ,(virgola) (1). 105
60 Carattere ,(virgola) (2). 105
61 Carattere $. 106
62 Selezionare l'elemento colonna di una variante. 107
63 Confronto tra due espressioni con operatore logico AND. 109
64 Confronto tra due stringhe con << 110
65 Confronto tra due stringhe con == 110
66 Divisione di due valori (1). 111
67 Divisione (2). 111
68 Appartenenza di un elemento ad un insieme (1). 111
69 Appartenenza di un elemento ad un insieme (2). 111
70 Comando RN0n1n2. 112
71 Presenza di una determinata variante. 112
72 Comando *#LET* (1). 114
73 Comando *#LET* (2). 114
74 Comando *#LET* (3). 115
75 Comando *#LETS* (1). 115
76 Comando *#LETS* (2). 115
77 Comando *#LETS* (3). 115
78 Comando *#LETS*, assegnazioni multiple. 116
79 Comando *#LETT* (1). 116
80 Comando *#LETT* (2). 116
81 Comando *#LETV*. 116
82 Comando *#LETS_MACROFINALE*. 117
83 Comando *#IF-#ELSEIF-#ELSE-#ENDIF* (1). 117
84 Comando *#IF-#ELSEIF-#ELSE-#ENDIF* (2). 118
85 Comando *#IF-#ELSEIF-#ELSE-#ENDIF* (3). 119
86 Comando *#IF-#ELSEIF-#ELSE-#ENDIF* (4). 119
87 Comando *#IF*. 119
88 Verifica se una variabile è definita, *#IFDEF*. 120
89 Comando *#ELSEIFDEF*. 120
90 Comando *#FOR* (1). 121
91 Comando *#FOR* (2). 122
92 Comando *#FOR* (3). 122
93 Cicli *#FOR* annidati. 122
94 Ciclo *#DO...#LOOP (#continue) (#break)* (1). 123
95 Ciclo *#DO...#LOOP (#continue) (#break)* (2). 123
96 Comando *#CONTINUE*. 124
97 Comando *#SQL*. 124

98 Costrutto *#SELECT*. 125
99 Comando *#MSG*, che visualizza il testo. 125
100 Comando *#MSG*, che visualizza il valore di una variabile. . . . 125
101 Comando *#INP* all'interno di un IF. 126
102 Comando *#GOTO*. 126
103 Comando *#VB* (1). 127
104 Comando *#VB* (2). 127
105 Scorrere il risultato di una query. 128
106 Scorrere il contenuto di un file testuale. 129
107 Inserimento di una quota. 129
108 Differenziare vari tipi di quotatura. 130
109 Operazioni algebriche nelle quotature. 130
110 Creare le quotature senza il comando *!QUOTA*. 130
111 Forzare il verso. 132
112 Gestire delle linee altezze. 133
113 Traslare una o più tipologie e poi annullare lo spostamento. . . 133
114 Aggiungere o togliere punti significativi da quotare. 134
115 Inserire una nota su una riga specifica. 135
116 Dichiarare un articolo come fuori misura. 135
117 Informazioni sull'articolo. 135
118 Creare diversi collegamenti tra i box. 135
119 Collegare box simili su linee base. 140
120 Leggere un'immagine. 141
121 Definizione di una sagoma semplice. 143
122 Associazione della sagoma al disegno. 143
123 Utilizzo comando A nella costruzione della sagoma(1). 144
124 Utilizzo comando A nella costruzione della sagoma(2). 145
125 Utilizzo comando E nella costruzione della sagoma. 145
126 Utilizzo comando C nella costruzione della sagoma. 145
127 Estrusione. 146
128 Comando !SAGDUMP. 147
129 Utilizzo comando !SAGPERCX. 148
130 Comando !SAGXY. 148
131 Ingrandire il profilo della sagoma. 149
132 Comando !SAGCLIPX. 149
133 Comando !SAGCLIPY. 150
134 Comando !SAGCLIPR. 150
135 Comando !SAGCLIPT. 152
136 Utilizzo del comando !FRESA. 155
137 Due frese sovrapposte. 155
138 Utilizzo del comando !FRESA. 156
139 Utilizzo del comando !FRESA. 156
140 Creare una lamiera. 158
141 Specchiare la sagoma nell'asse X. 160
142 Specchiare la sagoma nell'asse Y. 160
143 Specchiare la sagoma in entrambi gli assi. 160
144 Utilizzo comando !SAGUNION(1). 160
145 Utilizzo comando !SAGUNION(2). 161
146 Utilizzo comando !SAGDIFF(1). 161
147 Utilizzo comando !SAGDIFF(2). 161
148 Spostare la sagoma. 162

149 Due pannelli sagomati. . . . 162
150 Utilizzo del comando !SAGSPOSTA. . . . 164
151 Definizione degli archi per la sagoma. . . . 166
152 Costruzione della sagoma 166
153 Utilizzo del comando !SAGPERCX per la costruzione dei fianchi sagomati. . . . 166
154 Utilizzo dei comandi !SAGCLIPX e !SAGCLIPY per la costruzione dei ripiani sagomati. . . . 168
155 Utilizzo dei comandi !SAGCLIPX e !SAGCLIPY. . . . 169
156 Inserimento della base, cappello e schiena del mobile sagomato. 169
157 Utilizzo del comando !SAGOFFSET per la costruzione del top. 171
158 Taglio del bordo posteriore del top. . . . 171
159 Costruzione dell'anta sagomata. . . . 171
160 Introdurre file DXF. . . . 180
161 Funzione LOOK. . . . 181
162 Funzione LOOKB. . . . 182
163 IF in linea. . . . 182
164 Funzione CASE. . . . 183
165 Funzione DIR. . . . 183
166 Funzione LEN. . . . 183
167 Funzione MID. . . . 184
168 Funzione MID incompleta. . . . 184
169 Funzione ESTRAI. . . . 184
170 Funzione INSTR. . . . 185
171 Funzione FORMAT (1). . . . 185
172 Funzione FORMAT (2). . . . 185
173 Funzione FORMATV. . . . 186
174 Funzione SQL. . . . 186
175 Funzione CODVAR. . . . 187
176 Funzione VARIANTE. . . . 187
177 Funzione WEEK. . . . 187
178 Funzione DATA. . . . 188
179 Funzione DIM. . . . 188
180 Funzione EURO. . . . 188
181 Utilizzare le variabili di sistema legate ai pulsanti presenti nella finestra di disegno della regola. . . . 190
182 Apportare delle modifiche alle impostazioni di base. . . . 193
183 Parametro SE. . . . 195
184 Assegnare parametri aggiuntivi. . . . 195
185 Profondità del piano. . . . 196
186 Variabile che riporta il numero di pagina attuale. . . . 199
187 Comando LI. . . . 204
188 Comando TX. . . . 205
189 Comando FONT. . . . 206
190 Comando WMF. . . . 206
191 Comando CODE25. . . . 207
192 Comando CODE39. . . . 208
193 Comando COLORPENNA. . . . 211
194 Comando COLORERIEMPIMENTO. . . . 212
195 Comando TIPORIEMPIMENTO. . . . 212
196 Comando SPESSORE. . . . 213

197 Comando ORIGINE. 213
198 Istruzione ARG. 214
199 Parametro /BXTOPFRONT dell'istruzione ARG. 219
200 Istruzione TOP. 223
201 Stampa. 226
202 Commando WALL. 227
203 Aggiungere delle tipologie di lato per visualizzare i rullers. . . . 229
204 Aggiungere marker. 232
205 Stampa PROVA_MARKER.stn. 233
206 Stampa PROVA_MARKER.stn. 233
207 Stampa PROVA_MARKER.stn. 233
208 Utilizzo della funzione $[ex 241
209 Lista di elaborazioni. 248
210 Struttura del file *Disegno.CFG* 248
211 File *Disegno.CFG* 248
212 Inserimento ferramenta durante l'esplosione. 250
213 File di configurazione .CFG. 252
214 Il contenuto della variabile $(_isTop). 253
215 Utilizzo di &(_isTop) nel *#IF*. 253
216 Funzione `dau_getimballo`. 264
217 Funzione `wrPannello2`. 269
218 Funzione `wrx2`. 290

1 Introduzione

Lo scopo di questo manuale è quello di guidare il progettista alla creazione di un proprio catalogo di prodotti utilizzando ***3CadEvolution*** nei suoi vari aspetti: la progettazione gestionale(anagrafica di articoli e gestione dei prezzi), la gestione delle varianti e relazioni, il disegno 3D, il controllo di gestione e di distinta base, il controllo delle macchine utensili.

1.1 Definizione dell'utente del prodotto

Il target di utenti a cui si riferisce questo manuale è composto da persone che in ***3CadEvolution*** trovano lo strumento per progettare e realizzare nuovi modelli parametrici e combinandoli formare un catalogo.

1.2 Come leggere il manuale

In questo manuale verranno spiegate tutte le funzionalità di progettazione disponibili in ***3CadEvolution*** e il loro possibile utilizzo dagli utenti finali precedentemente descritti. Prima di tutto si parlerà di versioni, installazione e requisiti del programma, per proseguire successivamente argomentando i singoli aspetti del prodotto, e finiremo concentrandoci sulle cose più particolari e su esempi di utilizzo.

Il manuale dispone di: un indice iniziale attraverso il quale è possibile accedere direttamente al contenuto di ogni sezione, un indice delle figure, un indice dei codici e un indice degli esempi. Questo approccio è stato ideato per facilitare l'apprendimento del software sia per coloro che preferiscono lasciarsi guidare dagli esempi che per quelli che invece si trovano più a loro agio visualizzando i listati del codice sorgente.

Convenzioni e caratteristiche di questo manuale:

- Le parole scritte in *italico* trovano una precisa definizione nel glossario presente nell'appendice del presente documento;
- i nomi dei tasti o bottoni del interfaccia grafica sono sottolineati, quando sono uniti con il segno + significa che bisogna premere i tasti contemporaneamente;
- i nomi dei cataloghi sono scritti in maiuscolo;
- i termini più importanti vengono evidenziati in grassetto.

2 Definizione del prodotto

3CadEvolution è uno strumento parametrico che permette di gestire tutte le varie fasi di lavoro del mobile (progettazione, gestione ordini, controllo macchine). Il prodotto porta sostanziali cambiamenti nei sistemi di gestione della produzione, nel controllo degli errori e nei metodi di produzione flessibile. ***3CadEvolution*** consente al progettista di registrare i passi logici dell'iter di ingegnerizzazione del prodotto, a partire dallo schema di massima elaborato dall'architetto-designer, che comprende misure, modularità, stile, varianti, materiali e colori.

In una certa misura, si tratta di un processo rovesciato, nel senso che il disegno del prodotto diventa subordinato alla sua impostazione. Infatti, riprendendo le linee guida del designer, si definiscono secondo una gerarchia logica, modelli, misure dominanti, varianti, famiglie di componenti, tipologie grafiche e per ciascuna di esse, le regole di montaggio geometrico. Inoltre, per i componenti, si definiscono le regole che ne controllano le lavorazioni, creando una base dati tecnologica, fonte di tutte le informazioni necessarie e sufficienti per il processo produttivo.

La creazione degli oggetti grafici avviene attraverso un macrolinguaggio, che consente di parametrizzarne il comportamento in funzione delle varianti, siano esse, dimensionali, morfologiche, di materiale, di finitura e colore, varianti che, soltanto quando verrà configurato il mobile saranno definite dall'operatore. Questo approccio vale per le tipologie caratterizzate da parametrizzazione e variabilità, mentre per gli oggetti complessi, ***3CadEvolution*** consente di importare oggetti tridimensionali da altri *CAD* e anche in questo caso, di applicarvi semplici regole di parametrizzazione dimensionale. L'uno e l'altro approccio consentono di fatto, di rappresentare in modo completo l'ampia casistica del mobile. Inoltre e in alternativa alla ricostruzione geometrica delle facciate (ante, cassette, forni, etc.), è possibile applicare delle immagini fotografiche che, opportunamente scomposte possono adattarsi alle varie misure in larghezza e in altezza, mantenendo invariati certi dettagli (spessori di cornici, raccordi, sagome, etc.).

Operativamente, ciò significa anzitutto compilare le diverse tabelle di base che riassumono la modularità del prodotto (misure, colori, forme, etc.) e le sue varianti. Di seguito si impostano le tipologie elementari e le macrotipologie con le quali si definiscono i parametri necessari per ogni tipo di componente in forma totalmente parametrica. Ad esempio per un oggetto, come un'anta, verrà definita una sola tipologia che verrà utilizzata in maniera indipendente dalle sue dimensioni, dalla finitura, dal tipo di maniglia, dal tipo di cerniera, etc. Nello stesso tempo viene data la possibilità di costruire un sistema per definire tutti i componenti necessari alla sua produzione (distinta base a formule), tutti i calcoli da effettuare per ottenere dei prezzi o variazione di prezzo (listino a formule), gli schemi di foratura per le parti interessate(job), la rappresentazione grafica tridimensionale.

3CadEvolution mette a disposizione del progettista un sistema di controllo immediato con simulazione tridimensionale di tutte le regole create.

I disegni possono essere memorizzati in vari formati compatibili con la maggior parte dei sistemi di progettazione.

In ***3CadEvolution*** , l'utilizzatore non può inventare da zero nuovi prodotti, ma può realizzare migliaia o milioni di combinazioni a partire da prodotti resi parametrici. E' compito del progettista creare un catalogo in cui siano esposti determinati parametri e controllare come le risposte date ai parametri vadano a influire su disegno, prezzo, distinta base, macchina ecc.

2.1 Terminologia del prodotto

Utilizzeremo di seguito i seguenti termini:

- *Regola:* oggetto configurabile che viene riferito a uno o più articoli di vendita del listino. La regola è l'oggetto parametrico e definisce le domande che devono essere poste all'operatore, le loro limitazioni e la loro sequenza, e gli script che creano il prodotto date specifiche risposte (valutazione);
- *Definizione: (O Gruppo di Definizione):* è l'elenco delle domande cui l'operatore deve rispondere per definire il comportamento della regola: le domande seguono diversi criteri di valutazione e successione e le risposte possono essere limitate o forzate quando sono associate a un articolo, ma va sempre posta la massima cura nella rappresentazione della definizione della regola: bloccare situazioni incompatibili alla gestione evitando che l'operatore possa scegliere cose impossibili e rendere semplice e efficace la presentazione delle domande;
- *Valutazione: (o gruppo di valutazione)* è lo script di partenza per ottenere il disegno 3D, la distinta base: si basa sull'interpretazione delle risposte imposta nel gruppo di definizione; lo script può essere molto semplice e essenziale o complesso e annidato in funzioni e sottofunzioni(chiamiamo le funzioni Gruppi di valutazione o Gruppi);
- *Varianti:* sono le singole domande cui un operatore deve rispondere nei gruppi di definizione, scegliendo tra una e più opzioni;
- *Opzioni:* la singola risposta a una domanda individua e controlla la valutazione e le varianti successive, la presentazione delle opzioni è legata a differenti schemi (varianti di tipo diverso) e modi di essere visualizzate (classi di visualizzazione);
- *Tipologia elementare (o tipologia):* la riga base dell'elemento grafico da inserire, formata da una serie di attributi che permettono di accedere direttamente alle funzioni di disegno, di calcolo dei componenti, e di calcolo di maggiorazioni di prezzo.

2.2 Requisiti

Configurazione minima:

- Sistema operativo: Microsoft Windows Xp SP3, Windows Vista, Windows 7(64 bit è supportato). Per le versioni Windows 2000, Xp, Xp Sp1, Xp Sp2 occorre installare le DirectX 9.0c, Windows Internet Explorer 6.5 e verificare la presenza di MDAC 2.8. Non sono supportati Windows 3xx, Windows Me, Windows98;
- Processore: Pentium IV o corrispettivo AMD;
- Memoria RAM: 512 MB;
- Scheda video: 128 MB memoria video, compatibile 3D con shader 2.0;
- Altro: porta *USB* per la chiave di protezione HW.

N.B:	Tutte le macchine nuove, anche le peggiori, montano schede 3D con shader 3.0 e almeno 512 MB RAM. Bisogna risalire a prima del 2002 per trovare macchine che si prestano a dubbi in questo senso o a schede grafiche con chipset integrato.

Configurazione consigliata:

- Sistema operativo: Microsoft Windows Xp SP3 Windows Vista (64 bit è supportato). Per le versioni Windows 2000, Xp, Xp Sp1, Xp Sp2 occorre installare le DirectX 9.0c e Windows Internet Explorer 6.5. Non sono supportati Windows 3xx, Windows Me, Windows98;
- Processore: Pentium IV o corrispettivo AMD;
- Memoria RAM: 1 GB;
- Scheda video: 512 MB memoria video, compatibile 3D compatibile con shader 3.0;
- Altro: porta *USB* per la chiave di protezione HW.

Elenco schede video consigliate su (vedi le scelte del mese):
http://www.tomshw.it/graphic.php .

Nel caso in cui si debba utilizzare ***3CadEvolution*** su una macchina non connessa in rete, visto che il motore del database è abbastanza avido di memoria si consiglia l'utilizzo di un PC con almeno 1GB di RAM e possibilmente biprocessore.

3CadEvolution richiede l'installazione di un Database Server SQL. Sono supportate per questo sia le versioni gratuite (MSDE e SQL Express), sia versioni professionali del database Microsoft SQL server (MSSQL), versione 2000 e successive.

2.3 Installazione del programma

Vi sono tre diversi tipi di installazione del programma:

1. L'installazione server o stand alone: che installa tutto l'applicativo per l'utilizzo completo.
2. L'installazione del *Client* per un PC aggiuntivo collegato in rete a una precedente installazione di ***3CadEvolution*** : per permettere a più utenti di usare il programma sfruttando un database comune.
3. L'installazione ***eCadLite*** .

Prima di installare il programma occorre accertarsi di:

- Disporre del dispositivo di protezione hardware (chiavetta o dongle),
- disporre di sufficiente spazio su disco rigido (15 MB per il *Client*, più una misura variabile dipendente dai cataloghi: 500 MB media per catalogo),
- disporre dei diritti di amministrazione per installare sul PC (in particolare quelli con Vista e ***Windows 7*** [1]).

Inserendo il CD di installazione si ha una schermata visibile in Figura 2.1.



Figura 2.1: Loader dell'installazione di ***3CadEvolution*** .

Una volta caricato il loader viene eseguito in automatico il controllo dei prerequisiti necessari all'installazione.

[1] Nel caso di installazioni con sistemi operativi come Vista e ***Windows 7*** , lanciare il *Setup* dal CD cliccando col tasto destro sull'eseguibile e lanciandolo in modalità: Esegui come amministratore.

Sulle macchine con ***Windows2000*** non è installato ***Direct 9.0C***, in questo caso occorre installarlo premendo l'apposito pulsante che avvia la procedura di installazione standard ***Microsoft***.

Questo file, distribuito con il CD di installazione è di proprietà Microsoft e integra il sistema operativo con ***DirectX*** . Può sorgere il caso in cui il sistema non sia in grado di installare questo aggiornamento essenziale: in questo caso consigliamo l' *Upgrade* del sistema operativo verso XP ,Vista o ***Windows 7*** , magari con la supervisione di un esperto. Non è necessaria questa installazione nei sistemi Vista e XP, in quanto questo componente è già installato di serie, come si può verificare lanciando l'utility dxDIAG vedi Figura 2.2.

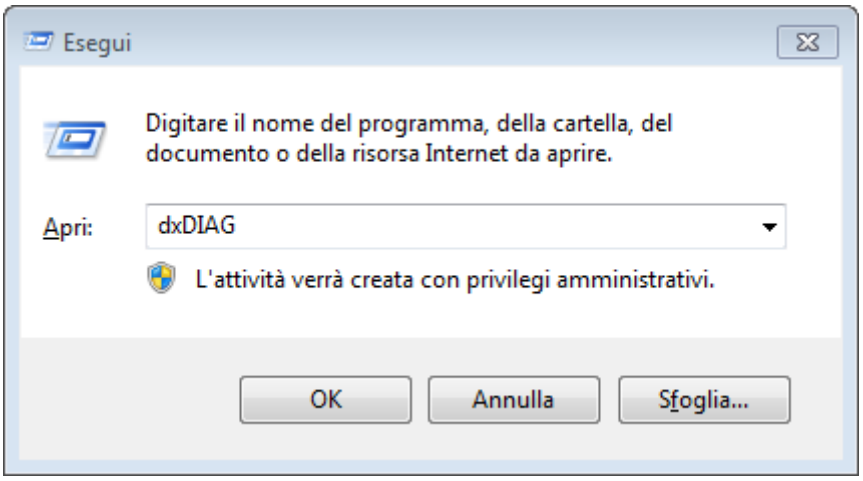


Figura 2.2: Lancio dell'utility dxDIAG.

Apparirà una scheda come in Figura 2.3 nella pagina 7 , dove sono riportate tutte le informazioni principali del sistema in uso, e in particolare, nella seconda pagina quelle relative a ***DirectX*** .

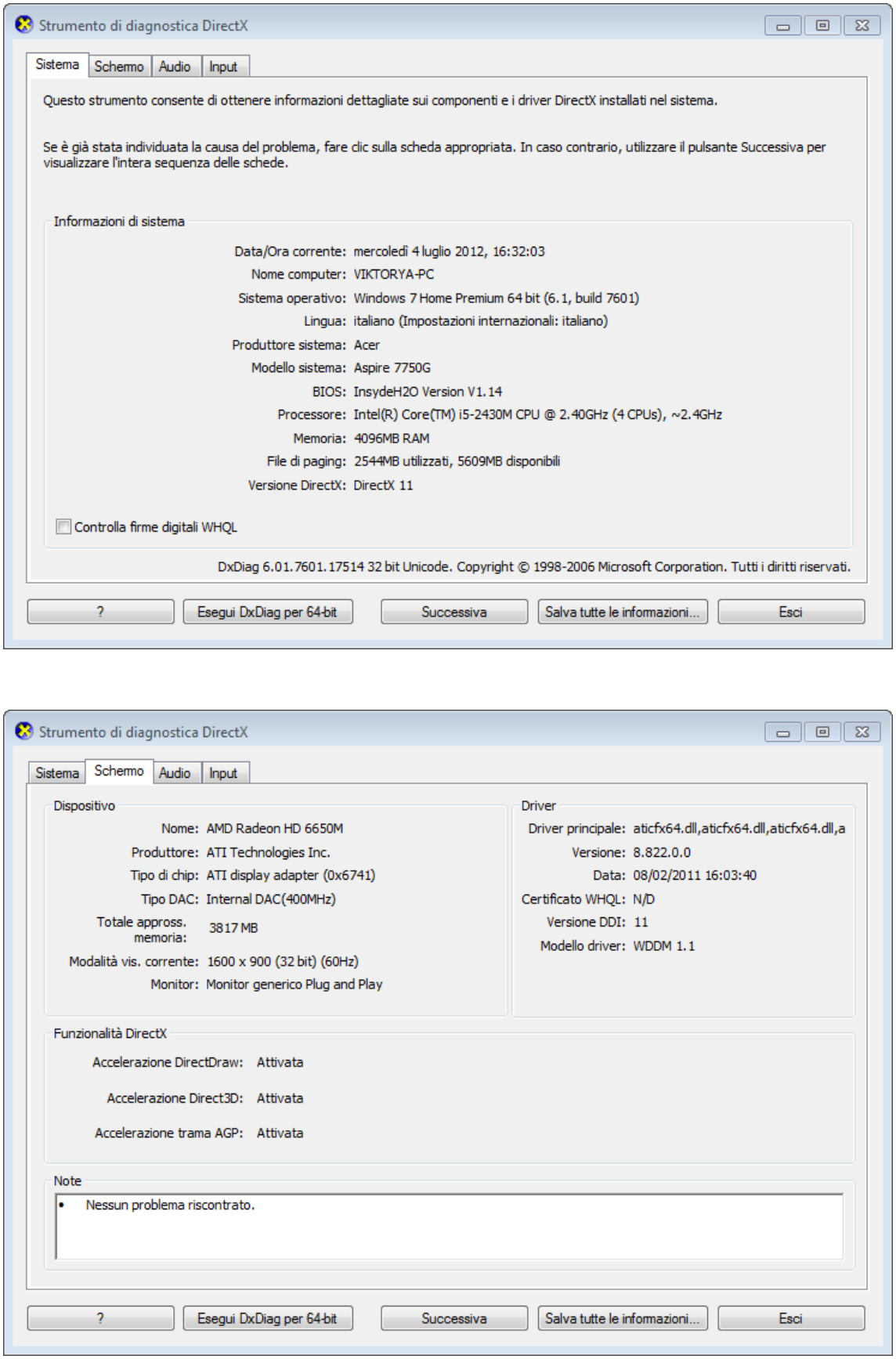

Strumento di diagnostica DirectX
Sistema
Schermo
Audio
Input
Questo strumento consente di ottenere informazioni dettagliate sui componenti e i driver DirectX installati nel sistema.
Se è già stata individuata la causa del problema, fare clic sulla scheda appropriata. In caso contrario, utilizzare il pulsante Successiva per visualizzare l'intera sequenza delle schede.
Informazioni di sistema
Data/Ora corrente: mercoledì 4 luglio 2012, 16:32:03
Nome computer: VIKTORYA-PC
Sistema operativo: Windows 7 Home Premium 64 bit (6.1, build 7601)
Lingua: italiano (Impostazioni internazionali: italiano)
Produttore sistema: Acer
Modello sistema: Aspire 7750G
BIOS: InsydeH2O Version V1.14
Processore: Intel(R) Core(TM) i5-2430M CPU @ 2.40GHz (4 CPUs), ~2.4GHz
Memoria: 4096MB RAM
File di paging: 2544MB utilizzati, 5609MB disponibili
Versione DirectX: DirectX 11
Controlla firme digitali WHQL
DxDiag 6.01.7601.17514 32 bit Unicode. Copyright © 1998-2006 Microsoft Corporation. Tutti i diritti riservati.
?
Esegui DxDiag per 64-bit
Successiva
Salva tutte le informazioni...
Esci
Strumento di diagnostica DirectX
Sistema
Schermo
Audio
Input
Dispositivo
Nome: AMD Radeon HD 6650M
Produttore: ATI Technologies Inc.
Tipo di chip: ATI display adapter (0x6741)
Tipo DAC: Internal DAC(400MHz)
Totale appross. memoria: 3817 MB
Modalità vis. corrente: 1600 x 900 (32 bit) (60Hz)
Monitor: Monitor generico Plug and Play
Driver
Driver principale: aticfx64.dll,aticfx64.dll,aticfx64.dll,a
Versione: 8.822.0.0
Data: 08/02/2011 16:03:40
Certificato WHQL: N/D
Versione DDI: 11
Modello driver: WDDM 1.1
Funzionalità DirectX
Accelerazione DirectDraw: Attivata
Accelerazione Direct3D: Attivata
Accelerazione trama AGP: Attivata
Note
Nessun problema riscontrato.
?
Esegui DxDiag per 64-bit
Successiva
Salva tutte le informazioni...
Esci

Figura 2.3: Informazioni su componenti e driver ***DirectX*** installati nel sistema.

Dopo aver verificato la presenza dei driver ***DirectX*** si potrà procedere con l'installazione di ***3CadEvolution*** *Client*, cliccando sull'apposito link nel loader. All'avvio dell'installazione verrà richiesto il percorso contenente la base dati, vale a dire il percorso mappato di rete contenente la cartella -_ECADPRO e la cartella del catalogo, Figura 2.4.

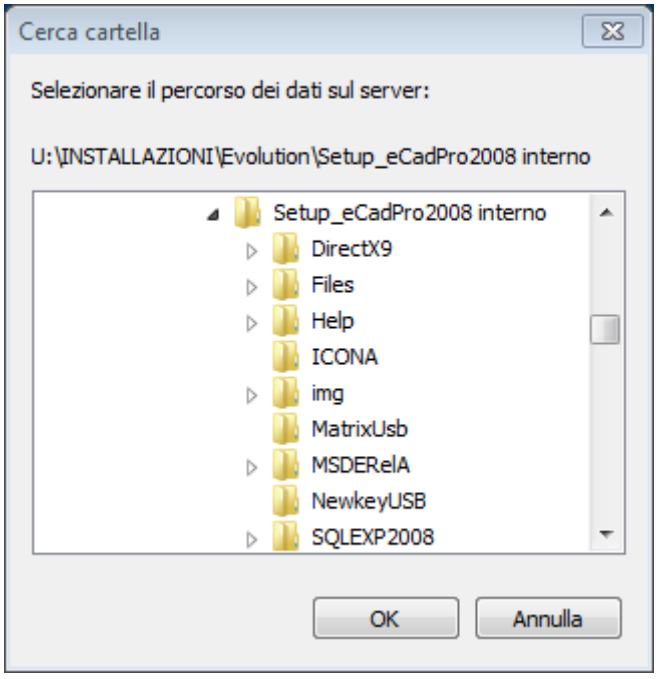


Figura 2.4: Selezione percorso dati.

1) Scegliere cartella di installazione valida (_ECADPRO).

Per installare ***3CadEvolution*** *Server*, occorre disporre di un'istanza *SQL*. Se l'istanza *SQL* è già presente in rete , il *Setup* propone la maschera UDL per la creazione della stringa di connessione al database, dove andranno specificati : l'istanza *SQL* a cui ci si vuole connettere e le relative credenziali di accesso, la schermata è visibile in Figura 2.5 nella pagina 9.

Se invece non si dispone di un istanza *SQL*, bisogna installarla. Il *Setup* prevede l'installazione di un'istanza di ***SQL Server*** Express 2008 R2, il link dell'installazione è visibile in Figura 2.6 nella pagina 9. L'installazione di questa istanza è vincolata alla presenza di due programmi:

- Microsoft Installer 4.5;
- Microsoft Net Framework 2.0 .

Se mancanti, il loader permette l'installazione dei due prerequisiti.

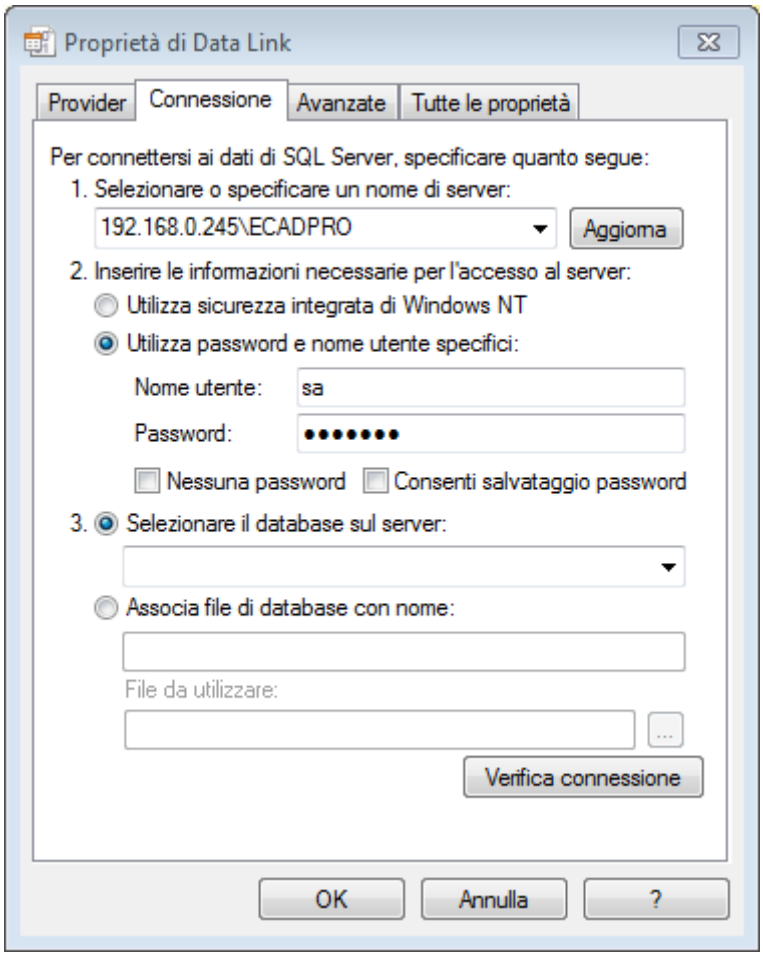


Figura 2.5: Universal data link.



Figura 2.6: Installazione ***SQL Server*** Express 2008 R2.

Inizio installazione: Dopo aver creato un istanza *SQL* per l'installazione *Server* o scelta la cartella necessaria per *Client* inizia l'installazione. In seguito sono elencati i passi che segue l'installazione del programma.

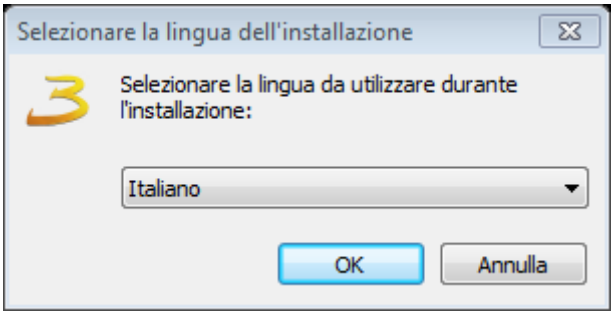


Figura 2.7: Selezione lingua.

1) Selezionare la lingua e fare click su Ok.

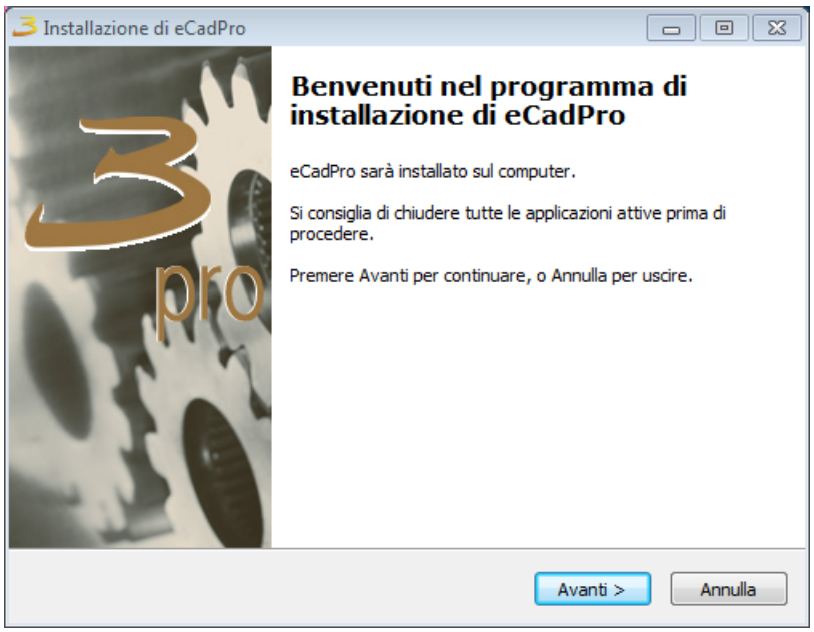

Installazione di eCadPro
Benvenuti nel programma di installazione di eCadPro
eCadPro sarà installato sul computer.
Si consiglia di chiudere tutte le applicazioni attive prima di procedere.
Premere Avanti per continuare, o Annulla per uscire.
Avanti >
Annulla

Figura 2.8: Inizio setup.

2) Dopo aver letto le informazioni fornite premere Avanti per proseguire con installazione.

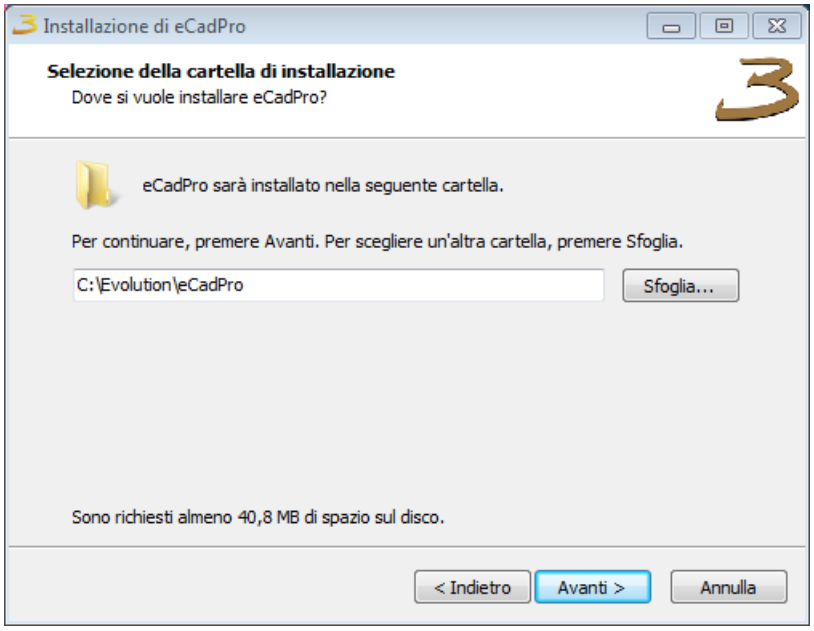

Installazione di eCadPro
Selezione della cartella di installazione
Dove si vuole installare eCadPro?
eCadPro sarà installato nella seguente cartella.
Per continuare, premere Avanti. Per scegliere un'altra cartella, premere Sfoglia.
C:\Evolution\eCadPro
Sfoglia...
Sono richiesti almeno 40,8 MB di spazio sul disco.
< Indietro
Avanti >
Annulla

Figura 2.9: Selezione della cartella di installazione.

3) Se si desidera installare il programma nella cartella proposta(la scelta consigliata) fare click su Avanti.

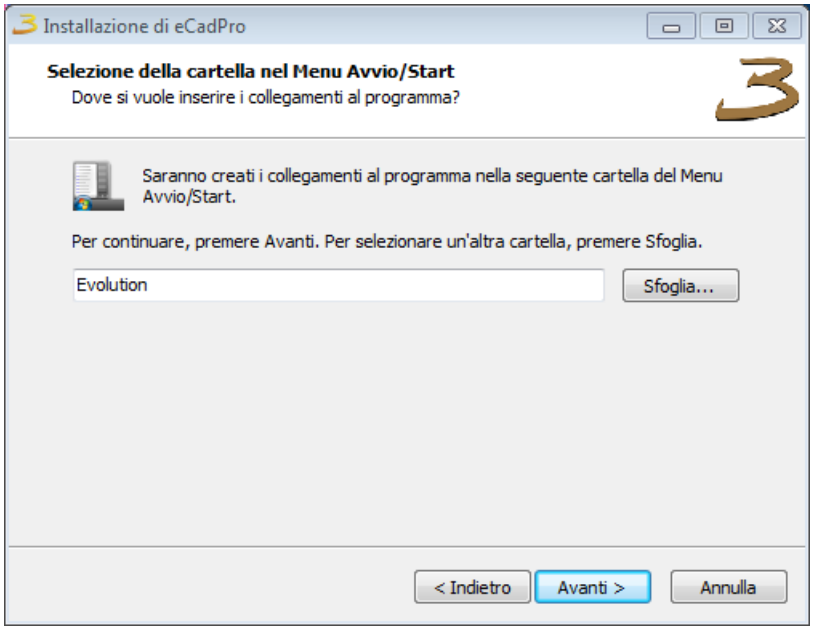

Installazione di eCadPro
Selezione della cartella nel Menu Avvio/Start
Dove si vuole inserire i collegamenti al programma?
Saranno creati i collegamenti al programma nella seguente cartella del Menu Avvio/Start.
Per continuare, premere Avanti. Per selezionare un'altra cartella, premere Sfoglia.
Evolution
Sfoglia...
< Indietro
Avanti >
Annulla

Figura 2.10: Selezione della cartella nel menu Avvio/Start.

4) Se si desidera utilizzare la cartella proposta nel menu Avvio/Start fare click su Avanti, altrimenti scegliere un'altra cartella.

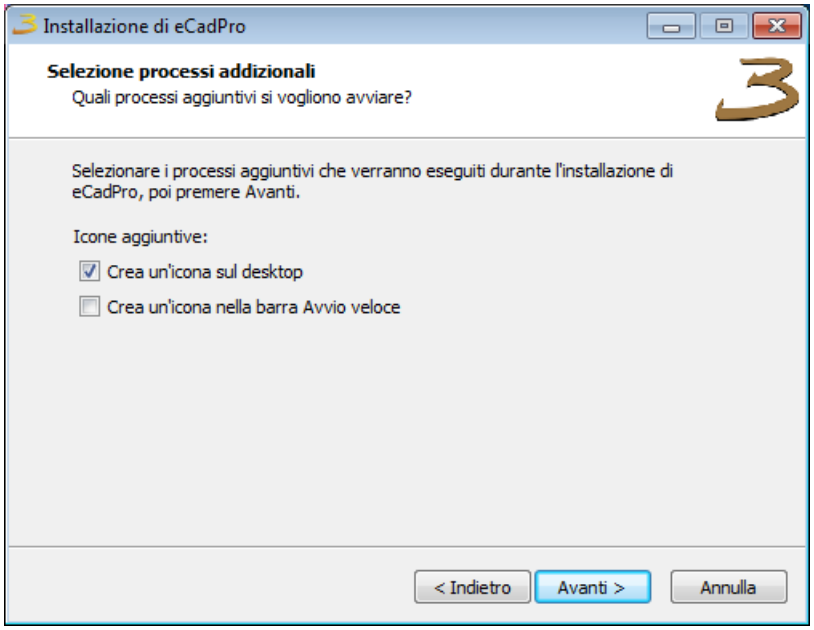

Installazione di eCadPro
Selezione processi addizionali
Quali processi aggiuntivi si vogliono avviare?
Selezionare i processi aggiuntivi che verranno eseguiti durante l'installazione di eCadPro, poi premere Avanti.
Icone aggiuntive:
Crea un'icona sul desktop
Crea un'icona nella barra Avvio veloce
< Indietro
Avanti >
Annulla

Figura 2.11: Creazione icone Desktop/Avvio veloce.

5) Se si desidera creare l'icona di ***3CadEvolution*** su Desktop o nella barra di Avvio veloce selezionare la scelta desiderata e fare click su Avanti.

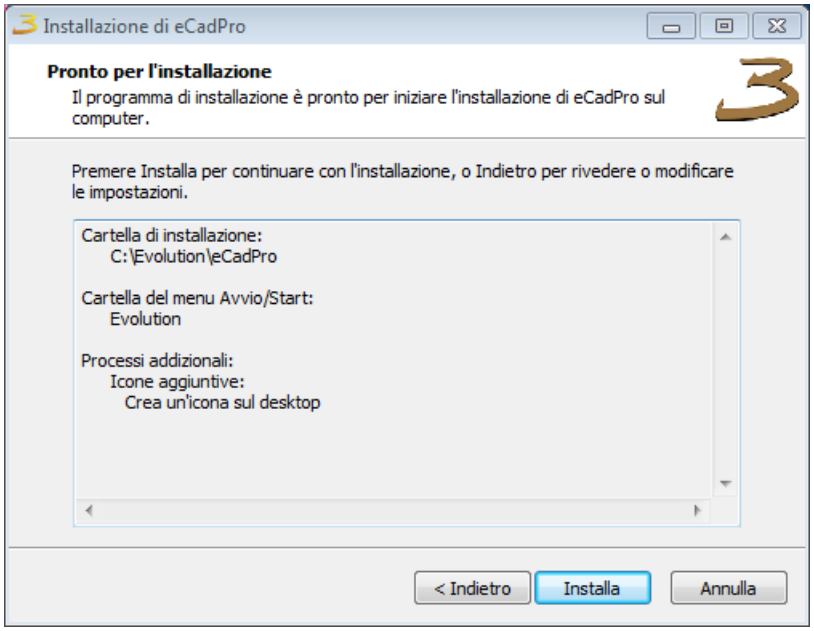


Figura 2.12: Avvio installazione.

6) Fare click su Installa per avviare installazione.

Dopo un'istallazione del programma tipo *Server* vengono effettuati in automatico:

- Ripristino dei file *BCP*;
- aggiornamenti del programma, presenti nella cartella files.

2.4 Installazione di un catalogo

Dopo aver installato il programma ***3CadEvolution*** vengono create 2 icone, la prima relativa al programma vero e proprio, la seconda per il programma ***eCadUtil*** , contenente le procedure di utilità per Creare, Cancellare, Copiare, Ripristinare, Parificare un catalogo e fare manutenzione al database.

Vediamo di seguito come utilizzare il programma di utilità ***eCadUtil*** .

Dopo aver lanciato il programma viene attivata la scheda visibile in Figura 2.13 (tranne nel caso in cui il programma non sia in grado di identificare il database, nel qual caso guarda sottosezione successiva):

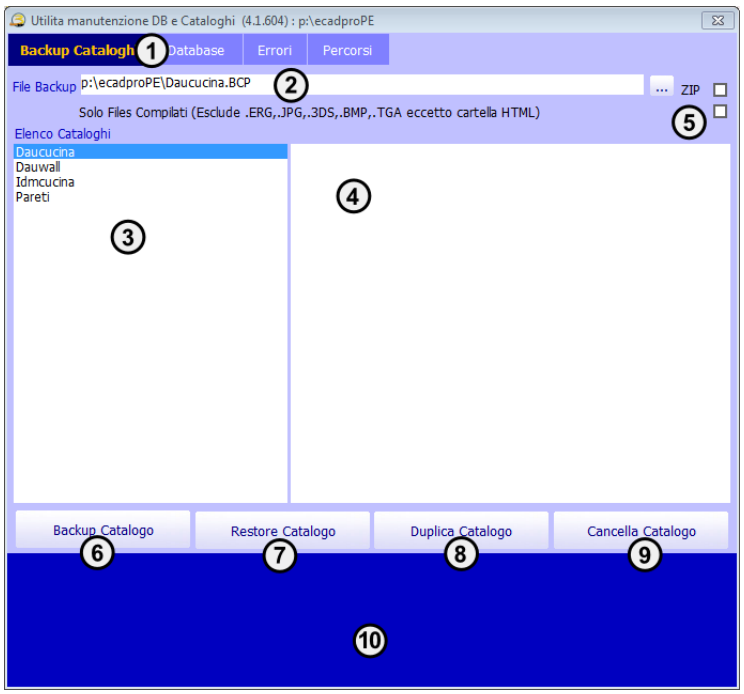


Figura 2.13: Installazione catalogo.

1. Come in tutte le parti dell'applicazione ***3CadEvolution*** , la finestra viene suddivisa in più schede (in questo caso 3: backup cataloghi, database e errori). Nel colore più scuro è evidenziata la scheda attiva.
2. Qui inseriamo il file di backup su cui intendiamo lavorare: i dati di un catalogo sono memorizzati all'interno di un unico file, questo file ha normalmente il nome del catalogo e estensione *BCP*, che viene generalmente memorizzato nella cartella di partenza in cui sono installati i cataloghi (default impostato in installazione c:\EcadPro).

3. Qui vi è l'elenco dei cataloghi installati e visibili dal programma + - Ecadmaster che non è un catalogo, ma il database su cui sono memorizzate le anagrafiche clienti, gli ordini e la produzione. ***3CadEvolution*** usa un database per ogni catalogo e memorizza tutte le informazioni del catalogo in una cartella che ha lo stesso nome del catalogo.
4. Il sistema utilizza quest'area per segnalare l'avanzamento delle attività di copia e ripristino.
5. Prima di effettuare il *Backup* selezionare questo check per copiare il catalogo soltanto in formato compilato. Questo preserva dalla possibilità di distribuire in versione sorgente i file delle regole e dei dati 3D. I dati saranno utilizzabili soltanto in forma compilata e non sarà possibile alterarli in alcun modo. E' in ogni caso una scelta dell'azienda distribuire il proprio catalogo in formato sorgente o solo compilato.
6. Dopo aver selezionato un catalogo nell'elenco (3) premendo questo tasto si attiva l'operazione di copia che può durare anche qualche minuto e che produce un file *BCP* contenente tutte le informazioni. Il nome completo del file è riportato in (2).
7. L'operazione di ripristino è il contrario dell'operazione di backup, cioè, a partire da un file *BCP*, selezionabile anche tramite ricerca su altri dispositivi. Per ripristinare un catalogo occorre che questo non esista su disco, per evitare accidentali sovrapposizioni di file. E' possibile naturalmente ripristinare un catalogo con un nome diverso, ma occorre fare attenzione ai preventivi e agli ordini inseriti che utilizzano il nome del catalogo come chiave di ricerca, se si cambia nome a un catalogo tutti i preventivi precedentemente fatti che utilizzino quel catalogo non saranno più disponibili. Per ripristinare una nuova versione di un catalogo sopra una versione esistente occorre prima cancellare il catalogo precedente.
8. Duplica un catalogo esistente creandone un copia precisa. Non si può duplicare un catalogo esistente, prima bisogna cancellarlo.
9. La cancellazione di un catalogo è completa e definitiva e lo rimuove tanto dal database quanto dalle cartelle che lo utilizzano. non vengono rimossi comunque gli ordini creati con quel catalogo. Fare attenzione e accertarsi sempre di avere copie aggiornate dei dati prima di una cancellazione.

Le funzionalità di manutenzione dei cataloghi sono piuttosto semplici da interpretare e impostare. ***3CadEvolution*** si basa su contenuti che sono all'interno dei cataloghi senza i quali il prodotto non può essere utilizzato, tali cataloghi possono essere autocostruiti o forniti dall'esterno. Queste procedure permettono di copiare, cancellare, rinominare e duplicare un catalogo.

Dalla versione 600, con il backup di ogni singolo database viene salvata una copia della struttura del database e di tutte le stored procedure del catalogo in due file:

- *ECADPRO.DFF*: data definition file con la struttura del database;
- *ECADPRO.SQL*: viste, *Stored procedures* e funzioni, per la rigenerazione.

Questi file vengono salvati nella sottocartella DB del catalogo, oppure nella sottocartella DB della cartella _ECADPRO per il database master(in questo caso il nome è ECADMASTER).

L'utilità di questi file è di poter creare un *Dump* per il controllo delle variazioni apportate manualmente alla struttura del database, pronto per essere comparato con altri database.

Inoltre la funzione legata al pulsante Esegui SQL file è stata estesa per permettere di eseguire sul database selezionato tanto file con estensione *SQL* , quanto comparare e modificare la struttura del database con i file DDF.

Questo dovrebbe facilitare il porting di variazioni alle strutture da un catalogo ad un altro e creare anche una documentazione delle stesse per controllo.

2.5 *eCadUtil* - Parametri da linea di comando

Per attivare le funzionalità automatiche di ripristino dei cataloghi o dei database sono disponibili in ***eCadUtil*** dei parametri gestibili da riga di comando.

/AUTO

parametro che serve per attivare la modalità automatica di ripristino dei *BCP* di ***3CadEvolution*** . Accetta come parametri l'elenco dei *BCP* da ripristinare separati dal carattere asterisco. I *BCP* specificati devono trovarsi all'interno della cartella di installazione del programma.

Esempio 1. Utilizzo del parametro /AUTO di ***eCadUtil*** .

/AUTO DAUCUCINA.BCP*DAUARREDAMENTO.BCP



Da usare solo per ***3CadEvolution*** .

/OFFLINE

questo parametro è usato durante l'installazione di eCadlite_Offline e serve per ripristinare in automatico il database eCadMasterss, i database cliente specificati e inserire all'interno di questi i parametri necessari per il corretto funzionamento del programma in modalità offline.

Esempio 2. Utilizzo del parametro /OFFLINE di ***eCadUtil*** .

/AUTO /OFFLINE DAUCUCINA;DAUARREDAMENTO

/SWAP

il parametro swap è necessario, sempre durante le installazioni offline di ***eCadLite*** , per specificare il nome programma che stiamo installando nel caso in cui quest'ultimo non coincida con il nome del catalogo.

Esempio 3. Utilizzo del parametro /SWAP di ***eCadUtil*** .

/AUTO /OFFLINE DAUCUCINA /SWAP DAU

2.6 Database

La seconda pagina del programma di utilità permette di lavorare con il database per effettuare delle manutenzioni.

Un catalogo è costituito di due parti: un database e una cartella contenente i file di lavoro del catalogo.

Nel capitolo precedente abbiamo visto come lavorare con l'intero database, mentre in questo ci concentriamo sulle attività di manutenzione del database, quali copiare e modificare i dati all'interno della cartella del catalogo è operazione che può essere fatta dall'utente direttamente utilizzando Windows Explorer.

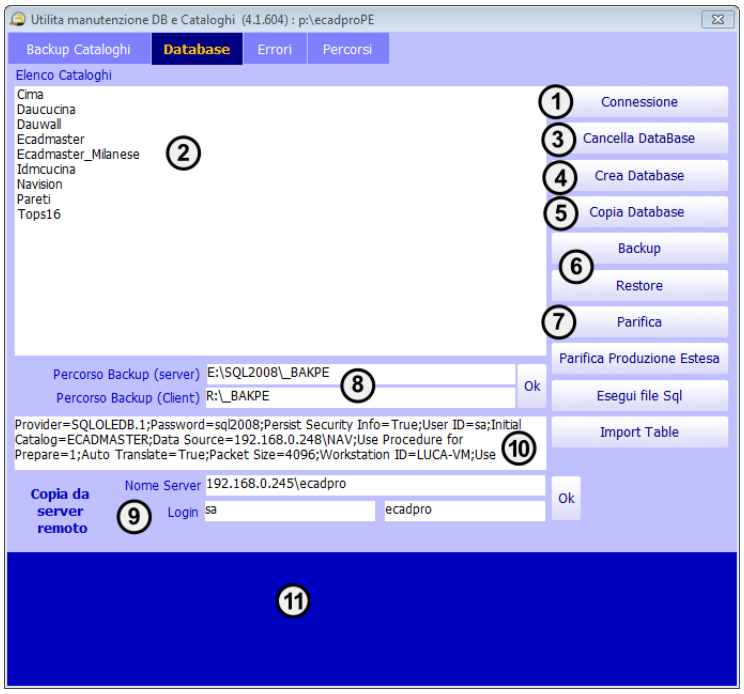


Figura 2.14: Manutenzione database.

1. *Connessione*: questa funzione permette di reimpostare la connessione con il database. Tale procedura si rende necessaria quando non sia mai stata impostata una selezione, in questo caso il programma ***eCadUtil*** da l'accesso solo a questa funzione, senza la quale non funziona niente altro, o quando si voglia cambiare il database.Premendola si accede alla funzione di connessione database standard. Nella prima pagina

visibile in Figura 2.14 si sceglie il provider OLEDB che deve essere sempre Microsoft OLE DB Provider from ***SQL Server*** , mentre nella seconda visibile in Figura 2.14 si sceglie il nome del server, nome utente e password. Premendo il tasto di verifica connessione il programma deve rispondere che la connessione è ok, altrimenti non si riuscirà a proseguire. In caso di installazione di MSDN o installazione locale il server di default è local , il nome utente è SA e la password è sa , ma i dati dipendono dall'installazione di *SQL* effettuata.

2. Se il programma è connesso si vedrà l'elenco dei database disponibili. ***3CadEvolution*** funziona se è stato creato il database ECADMASTER e se è installato almeno un catalogo.
3. Premendo questo pulsante si cancella il database selezionato (possono essere selezionati anche più database contemporaneamente).
4. Consente di creare un database azienda. Viene richiesto un nome per il database e quindi viene creato e parificato (vedi punto 7). Il programma considera Ecadmaster come nome particolare che va a creare con la propria struttura specifica. Per poter creare il database questo non deve essere già presente nell'elenco. Sia per la creazione di un database che di un catalogo valgono le regole sui nomi descritte precedentemente.
5. Un database viene duplicato con un altro nome; viene copiata la struttura, e tutti i dati.
6. Backup e Restore: corrispondono alle normali operazioni di copia database di ***SQL Server*** . Per chi non ne fosse pratico diremo semplicemente che il *Backup* prende tutti i dati di un database e li mette in un file [nomedatabase].BAK nella cartella indicata al punto 8, mentre il restore prende il file [nomedatabase].BAK e lo importa nel database. Questa è un'operazione molto importante soprattutto se si hanno dati sensibili all'interno del database, ma in realtà per ***3CadEvolution*** è comunque meglio fare le copie del catalogo utilizzando i file *BCP* . E' consigliata invece la copia di "ECADMASTER" dove sono contenuti gli ordini e i dati di produzione.
7. L'operazione di parifica del database è molto importante in quanto consente di effettuare automaticamente cambiamenti alla struttura del database nel passaggio da una versione del programma ad un'altra. Se tra una versione del programma e la successiva sono state integrate nuove tabelle, nuovi campi o modificata la struttura delle *Stored procedures* l'operazione di parifica consente di allineare i database in modo che la struttura sia conforme alle nuove gestioni. Nel manuale tecnico si parla dei file di parifica (*.DDF e *.SQL), ma diciamo qui da subito che è sconsigliabile modificare la struttura del database o di una procedura direttamente con le utilities di manutenzione del database. C'è infatti il rischio concreto che una successiva operazione di parifica vada a cancellare tutte le modifiche effettuate.
8. Percorsi: il *Backup* e restore del database vengono normalmente salvati in un percorso impostabile (il default è la sottocartella _BAK dell'installazione dei cataloghi). Qui è possibile modificare e impostare il percorso di salvataggio dei file di *Backup*. Tuttavia vi è da considerare che nel caso di una installazione di rete di ***SQL Server*** vanno dati due percorsi, che in realtà sono lo stesso però visti rispettivamente dalla macchina

su cui è installato fisicamente ***SQL Server*** e dal PC *Client* connesso. Ovviamente a causa di mappatura diversa del *Client* questi percorsi possono essere differenti.

9. Copia da altro Server: questa utilità consente di collegare una seconda connessione per copiare un database da un altro server: prima vanno dichiarate le caratteristiche di connessione, poi se connesso viene riempito l'elenco dei database e quindi abilitati i pulsanti per la copia da e verso altro database.
10. Area messaggi.
11. Area comunicazioni.

L'ultima cartella visibile in Figura 2.15 riguarda l'area errori e *Log* dove è possibile verificare eventuali errori e segnalazioni che il programma ha evidenziato eseguendo le varie operazioni.

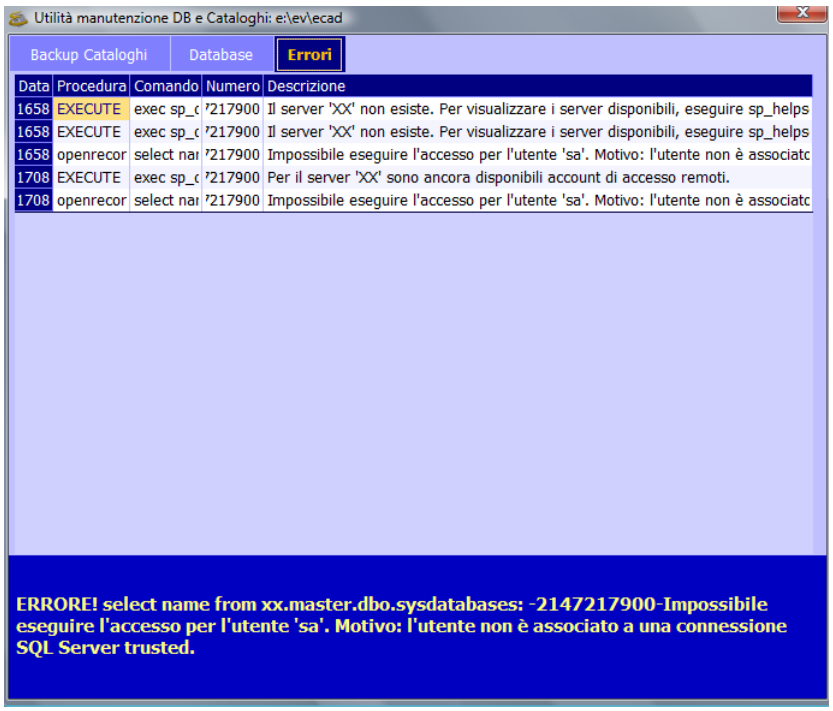

Utilità manutenzione DB e Cataloghi: e:\ev\ecad
Backup Cataloghi
Database
Errori
Data Procedura Comando Numero Descrizione
1658 EXECUTE exec sp_(7217900 Il server 'XX' non esiste. Per visualizzare i server disponibili, eseguire sp_helps
1658 EXECUTE exec sp_(7217900 Il server 'XX' non esiste. Per visualizzare i server disponibili, eseguire sp_helps
1658 openrecor select nai 7217900 Impossibile eseguire l'accesso per l'utente 'sa'. Motivo: l'utente non è associatc
1708 EXECUTE exec sp_(7217900 Per il server 'XX' sono ancora disponibili account di accesso remoti.
1708 openrecor select nai 7217900 Impossibile eseguire l'accesso per l'utente 'sa'. Motivo: l'utente non è associatc
ERRORE! select name from xx.master.dbo.sysdatabases: -2147217900-Impossibile eseguire l'accesso per l'utente 'sa'. Motivo: l'utente non è associato a una connessione SQL Server trusted.

Figura 2.15: Area errori e log.

2.7 Struttura dell'applicazione e caratteristiche

2.7.1 Organizzazione dei file

3CadEvolution gestisce tutti i dati all'interno della cartella in cui il programma viene installato.

Vi è una cartella per ogni catalogo e una cartella comune il cui nome è `_ECADPRO` e il cui contenuto è riportato in Figura 2.16 nella pagina 19.

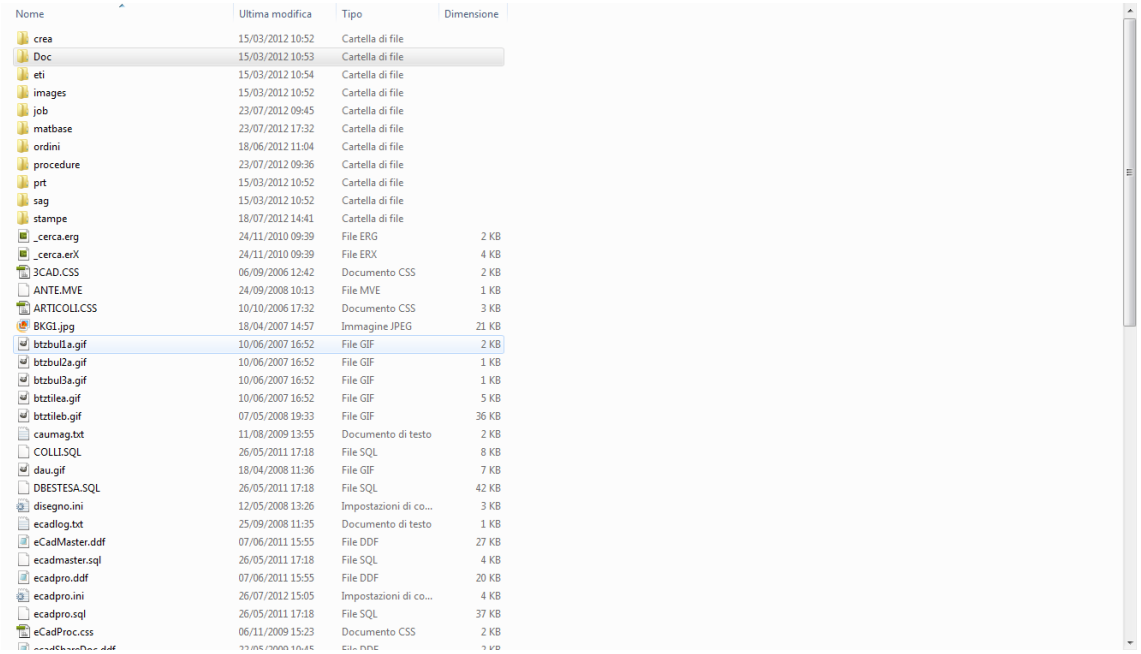

Nome
Ultima modifica
Tipo
Dimensione
crea 15/03/2012 10:52 Cartella di file
Doc 15/03/2012 10:53 Cartella di file
eti 15/03/2012 10:54 Cartella di file
images 15/03/2012 10:52 Cartella di file
job 23/07/2012 09:45 Cartella di file
matbase 23/07/2012 17:32 Cartella di file
ordini 18/06/2012 11:04 Cartella di file
procedure 23/07/2012 09:36 Cartella di file
prt 15/03/2012 10:52 Cartella di file
sag 15/03/2012 10:52 Cartella di file
stampe 18/07/2012 14:41 Cartella di file
_cerca.erg 24/11/2010 09:39 File ERG 2 KB
_cerca.erX 24/11/2010 09:39 File ERX 4 KB
3CAD.CSS 06/09/2006 12:42 Documento CSS 2 KB
ANTE.MVE 24/09/2008 10:13 File MVE 1 KB
ARTICOLI.CSS 10/10/2006 17:32 Documento CSS 3 KB
BKG1.jpg 18/04/2007 14:57 Immagine JPEG 21 KB
btzbul1a.gif 10/06/2007 16:52 File GIF 2 KB
btzbul2a.gif 10/06/2007 16:52 File GIF 1 KB
btzbul3a.gif 10/06/2007 16:52 File GIF 1 KB
btztilea.gif 10/06/2007 16:52 File GIF 5 KB
btztileb.gif 07/05/2008 19:33 File GIF 36 KB
caumag.txt 11/08/2009 13:55 Documento di testo 2 KB
COLLI.SQL 26/05/2011 17:18 File SQL 8 KB
dau.gif 18/04/2008 11:36 File GIF 7 KB
DBESTESA.SQL 26/05/2011 17:18 File SQL 42 KB
disegno.ini 12/05/2008 13:26 Impostazioni di co... 3 KB
ecadlog.txt 25/09/2008 11:35 Documento di testo 1 KB
eCadMaster.ddf 07/06/2011 15:55 File DDF 27 KB
ecadmaster.sql 26/05/2011 17:18 File SQL 4 KB
ecadpro.ddf 07/06/2011 15:55 File DDF 20 KB
ecadpro.ini 26/07/2012 15:05 Impostazioni di co... 4 KB
ecadpro.sql 26/05/2011 17:18 File SQL 37 KB
eCadProc.css 06/11/2009 15:23 Documento CSS 2 KB

Figura 2.16: Cartella _ECADPRO.

Le sottocartelle sono:

- CREA: contiene tutti i prototipi per la creazione di un nuovo catalogo e non è necessaria in fase di distribuzione del programma;
- DOC: contiene la documentazione tecnica, incluso il presente manuale
- IMAGES: contiene il riferimento alle immagini utilizzate dai vari file .HTML ;
- MATBASE: è una cartella fondamentale per l'uso del programma in quanto contiene le risorse e le traduzioni. Tale cartella non va modificata dall'utente;
- ORDINI: è la cartella di default per il salvataggio dei disegni;
- PRT: è la cartella di default per il salvataggio degli ambienti [2];
- SAG: contiene le sagome personalizzate e modificate attraverso il CAD2D;
- STAMPE: contiene i moduli utilizzati per le stampe;
- PROCEDURE: contiene le estensioni al programma.

Nella cartella principale sono contenuti diversi file di cui tracciamo un breve profilo:

- *.CSS: fogli di stile per le pagine *HTML*;
- *TESTA.ERG*, *TESTA.ERX*: regola di intestazione ordine. Contiene il riferimento ai vari campi necessari per la testata ordine(che può essere modificata a piacere in funzione delle esigenze gestionali);

[2] Gli ambienti non sono salvati insieme con gli ordini.

- *_CERCA.ERG*, *_CERCA.ERX*: regola per la ricerca degli ordini su database;
- *GENERAARTICOLI.SQG*: regola per il generatore di articoli;
- *ECADPRO.INI*: file di configurazione;
- *ECADPRO.DDF* e *ECADMASTER.DDF*: file di definizione del database di parifica per il database del catalogo e per il database di sistema. Questi file dovrebbero essere modificati solo dal produttore e sono rilasciate di volta in volta nuove versioni. La procedura di *Parifica Database* allinea il contenuto dei file DDF con la struttura reale del database;
- *ECADPRO.SQL* e *ECADMASTER.SQL*: completamento in sintassi *SQL* del database. Contiene la definizione di tutte le *Stored procedures* standard per il calcolo dei prezzi e per la gestione del database.

2.7.2 Cartella procedure

Tutte le estensioni al programma ***3CadEvolution*** sono memorizzate nella cartella procedure, contenuta nella cartella `_ECADPRO`.

L'indice per il richiamo di una procedura personalizzata è il file *master.htm* oppure *masterbase.htm*, che sono normali file in formato *HTML*, contenenti al loro interno dei link alle applicazioni.

Tali applicazioni di estensione possono essere dei file VBS (vbScript) oppure delle librerie compilate scritte con linguaggi diversi tra di loro.

Alcune estensioni richiamano funzioni interne di ***3CadEvolution*** , e pertanto sono accessibili direttamente via *Iperlink*.

La struttura per gli *Iperlink* è la seguente:

DAUINT: <NOMEPROCEDURAINTERNA> ,<PARAMETRI>

Oppure:

DAUPROC: <NOMEVBS>/<NOMEDLL>, <PARAMETRI>

N.B.: Sulle modalità per la scrittura di una *DLL* di estensione, consigliamo di rivolgersi direttamente al produttore.

L'elenco delle procedure interne è il seguente:

- SETUP: mostra la finestra del *Setup* del programma;
- REGOLE: attiva l'editor delle regole;
- MACRO: attiva l'editor delle macro;
- MACROTOP: attiva l'editor delle macro di tipo top;

- VARIANTI: attiva l'editor delle varianti. Come parametro accetta il nome di una variante , in questo caso permette di lavorare solo con quella variante specifica;
- TABELLE: attiva l'editor delle tabelle del catalogo, accettando come parametro il nome della tabella;
- TABMASTER: attiva l'editor delle tabelle presenti nel database ECADMASTER. Funziona come tabelle;
- TIPOLOGIE: attiva l'editor delle tipologie della regola;
- ASSOCIA: attiva l'editor delle associazioni;
- SPECIALI: finestra speciali di configurazione;
- SAGOME: attiva l'editor delle sagome;
- SQL: elaborazione di un comando *SQL* che crea una vista logica modificabile. Serve per una gestione rapida di interrogazione e modifica dei dati;
- CREAMINCAT: predispone la finestra per la ricostruzione delle miniature di un catalogo;
- CREAMINCOL: predispone la finestra per la ricostruzione delle miniature dei materiali di un catalogo;
- VIEW: elaborazione del comando *SQL*(come SQL), ma senza possibilità di modificare il risultato;
- BACKUP: backup del database del catalogo corrente;
- BACKUPS: backup del database ECADMASTER;
- RESTORE: ripristino del database del catalogo corrente;
- RESTORES: ripristino del database ECADMASTER;
- GENERA: attiva il generatore di codici articolo.

Per i nomi delle procedure esterne invece non vi è limite. Il programma comunque fornisce ad esempio le seguenti *DLL* :

- ANAGRAFICA.DLL: gestisce anagrafiche, articoli, barre, imballi, clienti;
- CREADBSISTEMA.DLL: creazione di una nuova ditta e importazione dati da 3CAD;
- MODELLI.DLL: gestione dei mobili;
- TABCONTATORI.DLL: gestione dei contatori;
- TABLIS.DLL: associazione listini per catalogo;
- CARICHI.DLL: gestione della produzione per carico;
- SALVADATABASE.VBS: salvataggio su database di un ordine e/o preventivo.

Queste *DLL* e VBS possono essere comunque sostituite, integrate in tutto o in parte per ogni installazione. E' attraverso queste *DLL* che il programma viene esteso e personalizzato. Tali file inoltre non sono soggetti a modifiche con le nuove versioni.

Nell'ambiente ***3CadEvolution*** per internet non sono gestite ne le procedure interne ne quelle esterne, inoltre le procedure interne sono comunque soggette a controllo di abilitazione della chiave.

2.7.3 File di configurazione

Il file *ecadpro.ini* serve a definire dei parametri per l'impostazione del funzionamento del programma.

Non si tratta di un unico file, ma bensì di 3 file distinti per la gestione delle personalizzazioni:

- Il file che si trova nella cartella `Windows` e che imposta i dati dell'utilizzatore corrente;
- il file che si trova nella cartella comune `_ECADPRO`;
- il file che si trova all'interno del catalogo.

Se apriamo il programma ***Notepad*** (start + esegui + notepad), scegliamo dal menu file+apri e digitiamo :

```
c:\windows\ecadpro.ini
( cartella diversa se windows non è installato in c:\windows)
```

Avremo la schermata visibile in Figura 2.17.

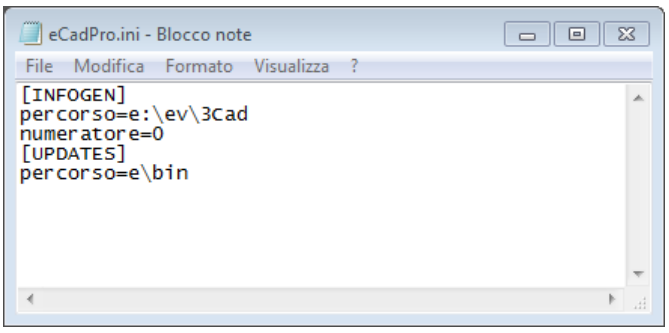


Figura 2.17: Schermata di ecadpro.ini

2.7.4 Numerazione degli ordini

In ***3CadEvolution*** ogni ordine/preventivo ha un riferimento principale numerico che corrisponde al numero dell'ordine.

Tale riferimento utilizza normalmente un allineamento di 6 caratteri, per cui l'ordine 1 viene indicato come 000001, ma in realtà possono essere utilizzati anche numeri più grandi di 999999. in questo caso l'allineamento è sulla dimensione del numero. Nel caso dell'ordine 1, supponendo di non aver modificato i percorsi di salvataggio di default messi nel file _ECADPRO \ECADPRO.INI, verranno salvati i seguenti file:

1. _ECADPRO \ORDINI \000001.EVE: file dell'ordine.
2. _ECADPRO \PRT \000001.PTR: informazioni sull'ambiente, il file di ambiente può essere chiamato anche diversamente dal file degli ordini, memorizzandolo o caricandolo dalle pareti con un altro nome.
3. _ECADPRO \PRT \000001.JPG: anteprima della parete.

   Possono poi essere creati altri file: se è previsto il salvataggio delle versioni un file per ogni revisione:
4. _ECAPRO \ORDINIREV \000001 \000001_xxx.EVE: notare una cartella per ogni ordine e un file per ogni revisione. Salvataggio delle bitmap per le etichette se prevista la collificazione automatica:
5. _ECADPRO \ETI \000001_xxxxx.JPG: dove xxxx è il numero di matricola del collo. Salvataggio delle lavorazioni:
6. _ECADPRO \JOB \000001_xxxx.ARG: o formato più compatto a seconda del sistema scelto per le lavorazioni.
7. _ECADPRO \JOB \T_000001.XML: formato delle lavorazioni per gli elementi lineari. Se nell'ordine vengono create sagome con l'editor delle sagome:
8. _ECADPRO \SAG\1_xxx.PTR .
9. _ECADPRO \SAG \1_xxx.EMF: dove xxx è il progressivo della sagoma.

Per questa ragione, il nome del file dell'ordine è elemento essenziale per mantenere tutti i collegamenti e pertanto non può essere rinominato come un qualsiasi file, ma deve essere modificato con le procedure previste dal programma. inoltre il salvataggio dell'ordine viene fatto anche su DATABASE, dove la parte gestionale dell'ordine è memorizzata nelle seguenti tabelle:

1. TOrdine: testata Ordine.
2. TOrdineRev: testata della Revisione.
3. TOrdineS: testata Sottoambienti.
4. BOrdine: distinta colli.
5. rOrdine: Righe dell'ordine.
6. pORdine: dettaglio prezzi dell'ordine.
7. DistintaT: distinta di primo livello associata all'ordine .

Il numero dell'ordine viene assegnato tramite la *Stored procedures* in ECADMASTER DauGetNumeratoreOrdine (che può essere modificata per esigenze particolari).

Tale numero è unico per tutti gli operatori ed è provvisorio fino al primo salvataggio dell'ordine; al primo salvataggio il programma va a verificare la presenza di un file .EVE con lo stesso numero e se lo trova comunque propone di cambiare la numerazione al primo numero successivo vuoto.

Tutte le stampe possono essere fatte solo se l'ordine/preventivo è stato salvato.

E' prevista la procedura di importazione dei protocolli da un sistema informativo gestionale che assegna proprie numerazioni (purché siano numeri interi compresi tra 1 e 4000000000) in questo caso vi sono degli script e gestioni per impostare i protocolli.

2.7.5 Salvataggio dell'ordine (Flag)

Un ordine o preventivo viene salvato in più passaggi: per prima cosa viene creato il file EVE che contiene tutte le informazioni dell'ordine in formato binario. Sono esclusi dal file EVE le pareti (file PRT) , gli oggetti riferiti al catalogo stesso (file 3D e immagini e regole), le sagome modificate manualmente e i file di lavorazione ARG. successivamente viene lanciata la procedura SALVADATABASE.VBS, che può essere personalizzata e che trasferisce l'ordine su database.

Poi intervengono i flag presenti in ecadpro.ini del catalogo.

COLLICREA

Il flag collicrea assume valori binari es. collicrea=1+2+4.

1 crea etichette (solo se abilitato M10)

2 permette la modifica manuale dei colli

4 esclude il salvataggio delle bitmap di preview

CREAETICHETTEBATCH

se = 1 questo flag disabilita la creazione delle etichette al salvataggio dell'ordine (come il precedente 4).

Se l'ordine è confermato e il flag JOBCREA=1 vengono creati i dati per la lavorazione: i file .ARG e i file .XML per la lavorazione dei TOP.

Inoltre è possibile impostare la trasparenza delle etichette, su *ecadpro.ini* di _ecadpro:

```
[INFOGEN]
coloreevidenziatoetichette (default 0,255,0)
[ENGINE]
EtichettaTrasparente (valori 1 e 0, default 1)
```

Su *ecadpro.ini* del catalogo per impostare la vista delle etichette:

```
[INFOGEN]
vistaetichette (defautl -30,15,0)
```

2.7.6 Impostazione di aggiornamento automatico del programma

Inserendo in *ecadpro.ini* la voce *PercorsoAggiornamenti* si dice al programma dove si vuole localizzare una cartella (in rete) per gli aggiornamenti automatici.

In questa cartella saranno contenuti tutti i file *AGG_xxx.EXE* con cui normalmente vengono rilasciati gli aggiornamenti.

All'avvio il programma verifica se il sistema è sincronizzato con gli ultimi aggiornamenti caricati nella cartella altrimenti li esegue, memorizzando nel file *aggiornamenti.ini* il risultato dell'operazione. Il file *aggiornamenti.ini* viene configurato inizialmente con un gruppo [aggiornamenti] nel quale va inserito l'elenco dei file da controllare per gli aggiornamenti.

Ad Aggiornamento effettuato il programma scriverà un nuovo gruppo [UtenteAgg_xxxx] dove xxx è il nome del PC aggiornato, con data e ora e esito dell'aggiornamento effettuato. In questo modo, nelle installazioni di rete, basterà copiare gli aggiornamenti nella cartella predisposta e ogni volta che un utente lancerà il programma, questo verificherà la presenza di nuovi aggiornamenti e se il caso provvederà a installarli.

Esempio 4. File *aggiornamenti.ini*

```
[aggiornamenti]
1=Agg_ecadpro.exe
2=Agg_ext.exe
```

Il file va posizionato nella stessa cartella in cui si trovano i file di aggiornamento *AGG_xxx.exe*

N.B: Funziona per l'esecuzione da parte di diversi utenti in quanto per ogni utente viene creata una sezione all'interno del file *aggiornamenti.ini*. Importante è che il nome PC sia univoco. Funziona anche con agg_dati e qualsiasi altro agg.

3 Creazione di un nuovo catalogo

3.1 Partenza

Dopo aver avviato ***3CadEvolution*** con una chiave abilitata alla gestione delle regole (modulo:M02), selezioniamo il menu Procedure (*MASTER.HTM*) visibile in figura Figura 3.1, e da questo menu la voce Utilità->Creazione nuova azienda e quindi creazione Database Utente.

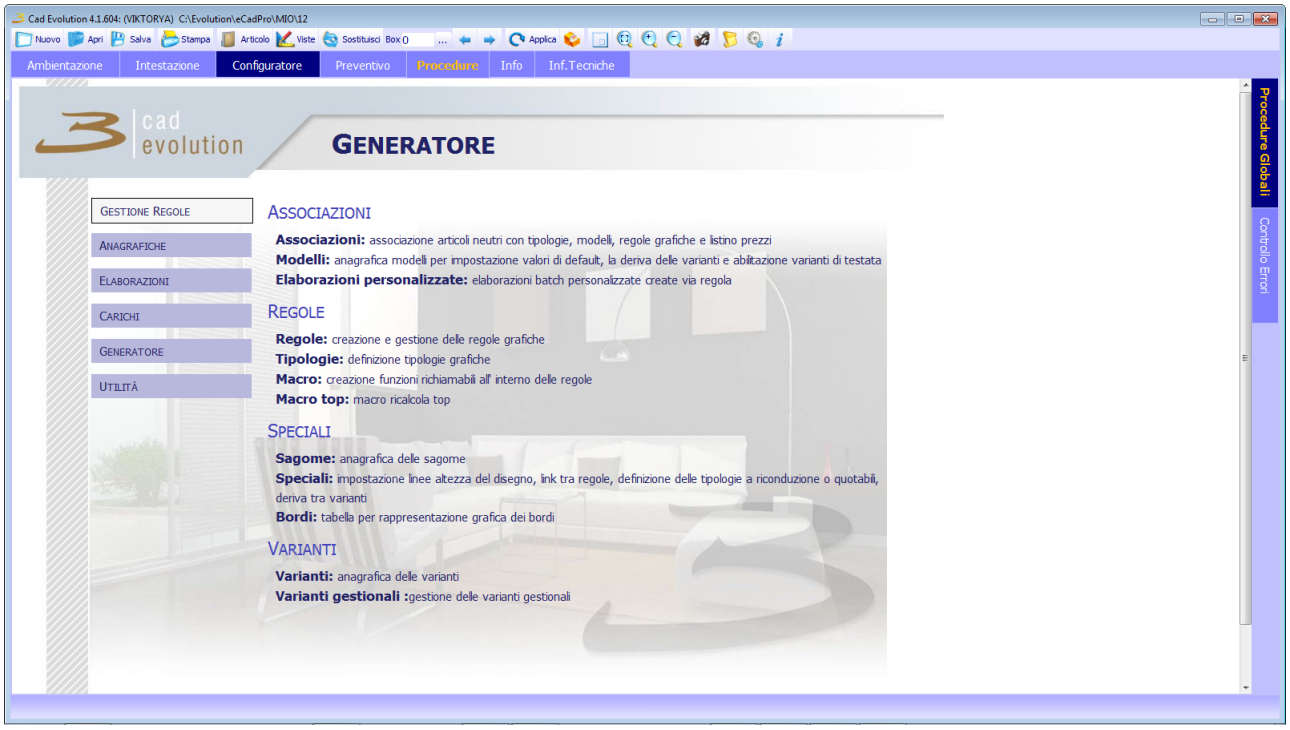


Figura 3.1: Menu procedure.

Qui viene richiesto il nome del catalogo nuovo come è possibile vedere in figura Figura 3.2 nella pagina 27. In questo esempio abbiamo chiamato il catalogo "MIO".

Alla pressione del tasto Procedi il programma ha:

- Creato un nuovo database MIO,
- parificato il database e caricato le *Stored procedures*,
- inserito i dati di base per lavorare, come: nella cartella `_ECADPRO CREA` vi sono i file BDR (vedi capitolo) per la creazione di dati su Database e l'impostazione di alcuni default;
- creato la cartella `mio` con le sottocartelle: `3ds`, `foto`, `html`;
- eseguito una compilazione iniziale di varianti e gruppi.

La parte più importante della procedura di creazione riguarda i default creati che sono diversi:

- Nella cartella `foto` il file *Bianco.DRG1*;



Figura 3.2: Inserimento nome catalogo.

- nella cartella del catalogo il file contenente gli script per le finiture *_epro_ante.vbs*, con l'impostazione;
- la regola di testata `GENE.ERG`;
- la regola di prova `001.ERG`;
- la tipologia di prova 001;
- i colori di prova 0,1,2,3;
- una variante *"COLORI"*;
- un articolo di prova 001;
- il modello per i prototipi 999.

In ogni caso quello che avviene creando un nuovo catalogo permette di poter agire immediatamente per la creazione di nuove regole e informazioni inserendo dei default modificabili.

Per verificare quanto detto, selezioniamo la ditta MIO e il modello 999 e inseriamo l'articolo di test. Dopo aver premuto Ctrl+I per la creazione della miniatura dell'immagine otterremo la finestra visibile in figura Figura 3.3 nella pagina 28.

Cosa è successo:

- Abbiamo creato un nuovo catalogo, in questo catalogo è stato inserito automaticamente un modello 999 che è il modello di servizio ossia il codice modello utilizzato per inserire tutti gli archetipi e le prove;
- abbiamo inserto un articolo neutro di prova 001 e lo abbiamo abilitato al modello 999 formando l'articolo composto 999_001;

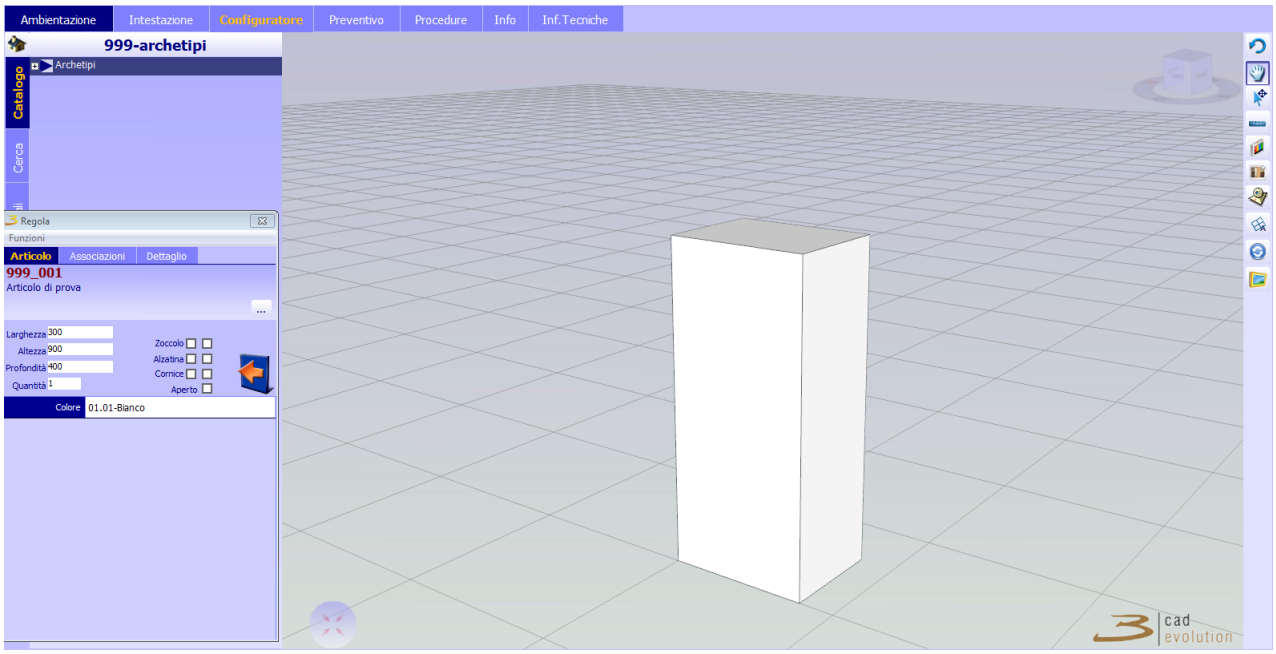


Figura 3.3: Articolo di test.

- abbiamo creato una regola `001.ERG` e l'abbiamo associata all'articolo neutro _001;
- questa regola è progettata per richiedere un colore da una variante apposita *"colore"* e a disegnare un box delle dimensioni dell'articolo del colore impostato con la variante;
- l'articolo è stato inserito nella classificazione 9900 sotto la voce prototipi.

Probabilmente tutto questo ora ha poco significato per il lettore, ma strada facendo chiariremo tutte le relazioni che hanno portato a questa impostazione e che sono meglio evidenziate nello schema seguente.

3.2 Creazione di un mobile componibile

Procederemo da questo momento per esempi, trasformando la regola che al momento disegna un solo BOX in una struttura più complessa.

Partiamo dall'interfaccia di modifica dell'articolo che vediamo in figura Figura 3.4 nella pagina 29.

Il menu funzioni permette di accedere alla modifica delle regole, alla gestione delle associazioni, alla gestione delle varianti e alla modifica della grafica (quest'ultima è semplicemente una scorciatoia per la modifica del gruppo di valutazione della regola).

Premendo Ctrl+G o scegliendo a menu la voce Modifica Grafica accediamo alla parte grafica della regola. Quello che vediamo inizialmente è una pagina vuota con un testo che è riportato di seguito:

```
001 // Test
```

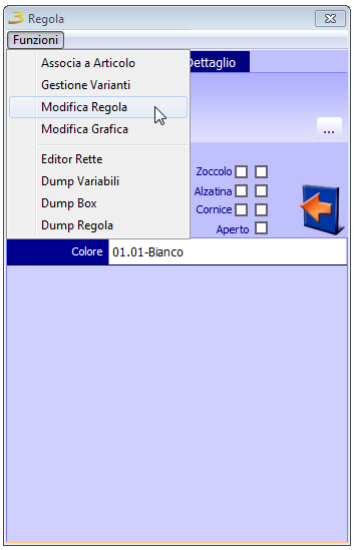


Figura 3.4: Modifica articolo.

001 sta a significare che il programma inserirà il componente 001, mentre tutto quello che segue i caratteri // è un commento.

Se passiamo alla scheda Dettaglio, dove vediamo e possiamo modificare manualmente l'articolo, vedremo che sono stati inseriti i seguenti dati come possiamo notare dalla Figura 3.5:

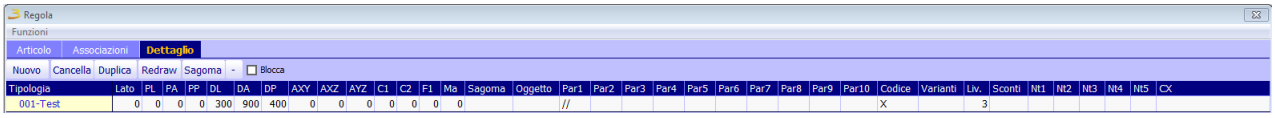


Figura 3.5: Dettaglio articolo.

A questo punto dall'editor delle regole posizioniamoci sulla riga che contiene la voce 001 e premiamo il tasto Ctrl+H, per entrare nello schema di modifica della regola visibile in figura Figura 3.6.

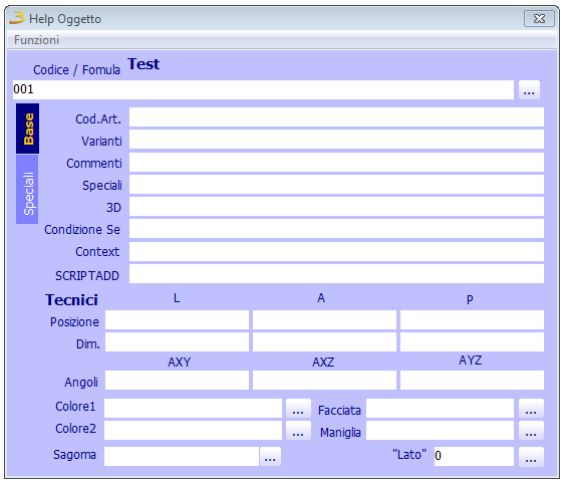


Figura 3.6: Schema di modifica della regola(Help Ogetto).

Posizioniamoci su Dim.L e scriviamo nella casella il valore 30. Premendo invio la finestra si chiude e il programma scriverà la riga:

Esempio 1. Cambio Dim.L attraverso una regola.

Codice 1. Assegnazione di un valore alla variabile L .

```
001, L=30 // Test
```

Chiudendo la finestra (Alt+F4) verrà interpretata e disegnata la nuova regola, dove il box è più sottile, per l'esattezza la dimensione impostata è di 30 mm.

Torniamo all'editor delle regole e copiamo la prima riga sulla seconda facendo in modo che il nostro script riporti:

Esempio 2. Due pannelli di spessore 30mm.

Codice 2. La stessa tipologia con parametri diversi.

```
001,L=30 // Test
001,sl=300,L=30 // Test
```

E usciamo dal programma: vedremo disegnati 2 pannelli (fianchi) di spessore 30 mm.

Ricapitoliamo:

- L'editor delle regole (o distinta delle regole) è una pagina di testo su cui si può scrivere liberamente;
- premendo Ctrl+H su ogni riga viene attivata una pagina di help che aiuta a compilare i vari parametri della riga;
- la struttura per disegnare un oggetto 3D è una riga composta da un numero di tipologia + una serie di parametri:
  - xxx : è un numero intero compreso tra 1 e 999 (possono esistere solo 999 tipologie diverse in un catalogo, salvo le sottotipologie);
  - p1=..., p2= ..., ... pn=...:sono i parametri che danno significato e comportamento all'oggetto, ad esempio *L* modifica la larghezza e *SL* lo spostamento in larghezza.
- viene disegnato un parallelepipedo come struttura di base;
- tutto quello che viene disegnato corrisponde a quanto si vede nella struttura della variazione di componenti.

3.3 Variabili

Facciamo a questo punto un esperimento con il codice, impostando una variabile per lo spessore dei fianchi. Ciò che vogliamo ottenere è che si possa agevolmente cambiare lo spessore senza intervenire sullo script.Per questo inseriamo le seguenti modifiche alla regola, digitandole semplicemente con l'editor:

Esempio 3. Assegnazione alla Dim.L del valore di una variabile attraverso una regola.

Codice 3. Variabile spessore.

```
#LET spessore=30
001,L=$(spessore)> // Test
001,sl=300,L=$(spessore) // Test
```

Nella prima riga vi è il comando di assegnazione di un valore alla variabile Spessore. Il comando è *#LET* anche se ne esistono diverse forme: *#LETS* , *#LETV*.

Le righe contenenti comando iniziano sempre con il carattere # che è appunto il prefisso dei comandi.

Scrivendo *#LET* spessore=30 abbiamo memorizzato nella variabile spessore il valore 30.

Una variabile è una parola che inizia con una lettera e contiene successivamente numeri o lettere:

- Alfa, Beta, A125, Aereo_01: sono nomi di variabili valide;
- A;34, 123AA, A*3: non sono nomi validi.

Avremo potuto anche scrivere:

Esempio 4. Assegnazione di un calcolo algebrico come valore di una variabile.

Codice 4. Calcolo algebrico come valore della variabile con *#LET*.

```
#LET spessore=5*4+(3+2)*2
```

Per ottenere lo stesso risultato, ossia assegnare 30 alla variabile spessore. Al di là dell'esercizio vedremo che la possibilità di lavorare con formule è fondamentale per lavorare con le regole.

Nelle righe successive abbiamo utilizzato invece $(spessore). Quando si utilizza una variabile al di fuori dell'assegnazione ,si vuole avere il suo contenuto, va scritta nella forma *$(<nome variabile>[:<colonna>])*.

Il contenuto tra parentesi quadre è opzionale! In buona sostanza una variabile è un testo contenuto tra parentesi tonde e con un $ come prefisso.

Esempio 5. Utilizzo delle variabili all'interno della regola.

Codice 5. Variabili all'interno di un calcolo algebrico assegnato come valore di una variabile.

```
#LET bordo=5
#LET interno = 18
#LET spessore=$(interno)+$(bordo)*2
```

Provate a cambiare il contenuto delle variabili ed esercitatevi sull'utilizzo.

3.4 Varianti

Il fatto di cambiare una variabile nel modo visto prima è interessante, ma non aiuta l'operatore che deve scegliere tra diversi spessori.

Per questo opereremo creando una variante spessore dove possano essere selezionati i seguenti valori : 12, 18, 25, 60.

Accediamo alla regola con il tasto Ctrl+R e otteniamo la finestra visibile in Figura 4.2 nella pagina 93:

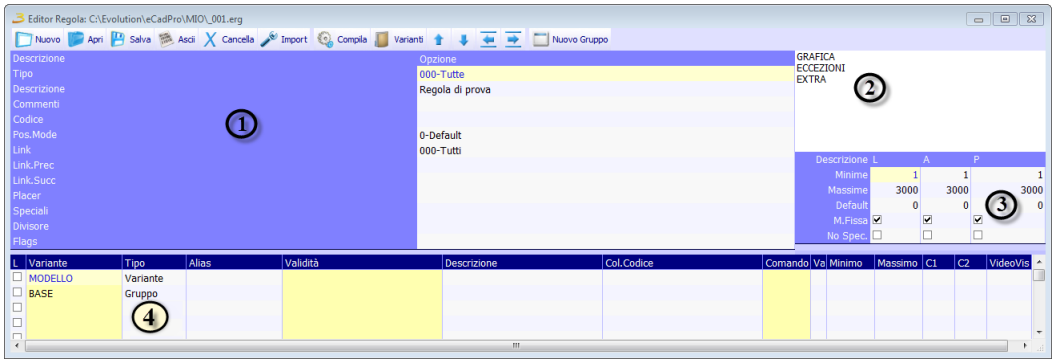


Figura 3.7: Editor regola.

Questa è l'intestazione e l'editor completo della regola. Prima di procedere cerchiamo di familiarizzare con questa interfaccia.

1. In questa area sono riportate le informazioni di intestazione della regola: gruppo di appartenenza, descrizione, gruppo per la definizione del collegamento con altri oggetti e altre informazioni; specificheremo meglio quando è il caso di modificare le impostazioni di default che di solito sono sufficienti.

2. Gruppi di valutazione: selezionando il gruppo GRAFICA vedremo il testo della regola precedente. Anche se questo è il gruppo più importante non è l'unico: il gruppo EXTRA fornisce le informazioni per il collegamento con i muri in caso di angolo e il gruppo di eccezioni permette di verificare quali condizioni di risposta non sono accettate dal programma. Possiamo inoltre creare nuovi gruppi in cui memorizzare copie dei dati a nostro piacere
3. Blocchi sulle dimensioni: questo imposta dimensioni minime e massime e restrizioni dell'oggetto. In realtà queste informazioni non sono essenziali in quanto le restrizioni possono essere applicate anche all'articolo direttamente
4. Definizione delle domande e risposte: questa è l'area principale di intervento per creare nuove varianti e organizzarle. La finestra è una griglia divisa in colonne.

Interveniamo quindi nell'area 4 per capire come è strutturata e come intervenire per le modifiche di questa tabella. Si tratta di una griglia in cui per ogni riga sono impostate le domande da fare all'operatore, divisa in colonne il cui significato è il seguente.

Lock: blocca la variante (non utilizzare se non per casi limite: lo stesso effetto viene raggiunto sulla regola con l'istruzione *#LETV*).

Nome della variante: riporta il nome della variante.

Tipo di variante: la variante è infatti impostata per essere utilizzata in diversi modi: vi sono varianti libere, varianti a selezione multipla, varianti si/no o no/si, separatori di campo, varianti nascoste (approfondiremo strada facendo).

Alias: è possibile accedere a una variante memorizzata con un nome attraverso un altro nome, creando un ALIAS.

Validità: è una formula che permette di impostare se la variante è visibile o non visibile, in genere è condizionata dal valore assunto dalle varianti precedenti.

Descrizione: pur essendo consigliato impostare la descrizione direttamente nella tabella delle varianti in questa colonna può essere reimpostata o immessa quando non è possibile. La descrizione è ciò che l'operatore vede selezionando la variante.

Col.Codice: formula per il calcolo della validità non per l'intera variante ma per ogni singola opzione: permette di limitare le opzioni selezionabili in base al contenuto di una scelta precedente: occorre però attenzione affinché almeno una opzione rimanga comunque valida.

Comando: script avanzato da eseguire alla selezione della variante; come nel caso del campo Lock evitare di utilizzarlo a meno che non sia necessario.

Valori: aprendo l'editor è possibile impostare i valori per la variante; normalmente questi valori non dovrebbero essere impostati nella regola ma nella tabella delle varianti (serve solo come eccezione).

Minimo: valore minimo selezionabile per un campo di tipo misura.

Massimo: valore massimo selezionabile per un campo di tipo misura.

C2:impostazioni del colore background e foreground con cui viene visualizzata la variante.

Dopo questa lunga premessa procediamo per creare una variante *"SPE"* in cui imposteremo gli spessori richiesti. Definiamo questa variante di tipo VARIANTE, come in Figura 3.8.

L	Variante	Tipo
☐	MODELLO	Variante
☐	BASE	Gruppo
☐	SPE	Variante
☐		

Figura 3.8: Variante che imposta gli spessori.

A questo punto inseriamo i dati accedendo alla voce VARIANTI(vedi Figura 3.9) nella toolbar e inserendo i valori che ci siamo prefissi (da notare che abbiamo impostato la descrizione = spessore e il default = 25).

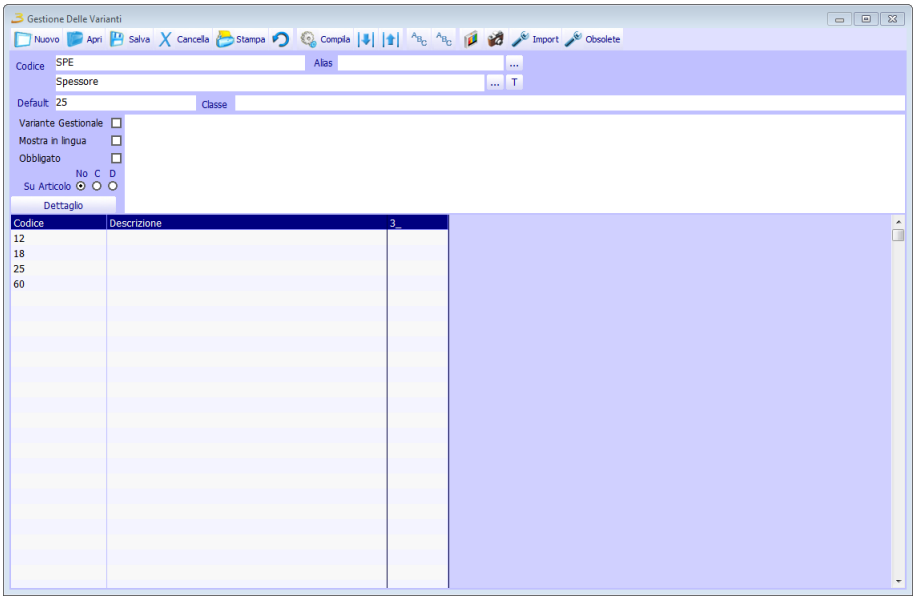


Figura 3.9: Editor varianti.

Dopo aver salvato la variante, torniamo alla grafica e modifichiamo la prima riga come segue:

Esempio 6. Variante spessore rappresentata nella grafica.

Codice 6. Variante spessore parametrizzata.

```
1 #LET spessore=$(spe:0) // Assegnazione dello spessore
```

A questo punto consigliamo il lettore a provare qualche modifica in autonomia.

Ancora varianti

Vogliamo modificare l'esempio in modo che l'operatore possa cambiare non la scelta dello spessore ma del tipo di mobile identificando un ipotetico mobile economico (spessore 18) uno standard (25) e uno speciale di spessore selezionabile dall'utilizzatore liberamente.

Per far questo modifichiamo la regola sostituendo alla variante *"SPE"* una variante *"Tipo"*.

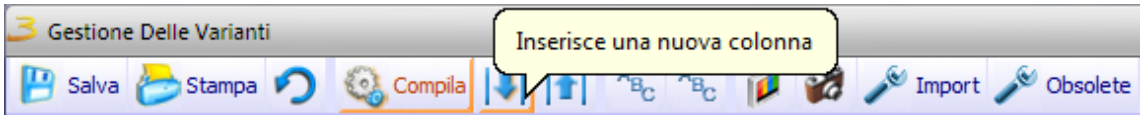


Figura 3.10: Inserimento nuova colonna.

Inseriamo una nuova colonna premendo il tasto sulla toolbar, e selezionando il titolo della colonna (doppio click) impostiamo il valore spessore e gli spessori per ogni voce, Figura 3.11.

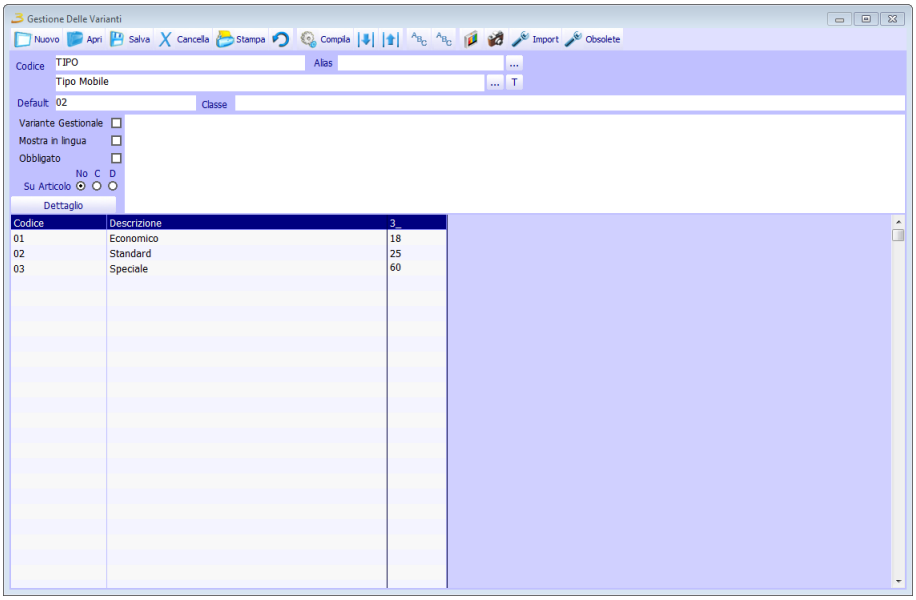


Figura 3.11: Variante spessore.

A questo punto modifichiamo la regola scrivendo:

Codice 7. Assegnazione della variabile spessore ai fianchi.

```
#LET spessore=$(tipo:3)
001,L=$(spessore) // Test
001,sl=300,L=$(spessore) // Test
```

Qui notiamo la differenza circa l'utilizzo della variante: l'operatore non seleziona direttamente lo spessore, ma il tipo di mobile e nella tabella varianti abbiamo inserito una colonna (la terza) in cui impostiamo lo spessore reale. Dunque la *variante* è una variabile divisa in colonne in cui è possibile accedere a ogni colonna attraverso il suo id numerico. E' possibile aggiungere e togliere colonne (ma solo alla fine) e referenziarne i valori.

Possiamo anche scrivere nella regola:

Esempio 7. Messaggio a video, visualizzazione del valore di una variabile.

Codice 8. Comando *#MSG*, messaggio a video.

```
1 #MSG $(tipo)
```

Questo mostrerà il contenuto della variabile (variante) *"Tipo"* in un messaggio di sistema come in Figura 3.12

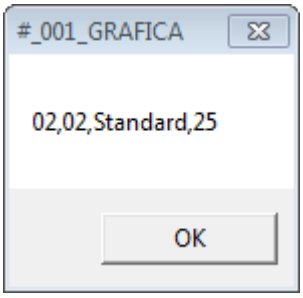


Figura 3.12: Visualizzazione a video del contenuto di una variabile .

Separati da [,] vediamo i contenuti di ogni colonna: notiamo che le colonne sono numerate da 0 in poi e che la colonna 0 ha lo stesso valore della colonna 1 (anche se questo non è sempre vero). La colonna 0 è quanto ha scelto l'operatore, mentre la 1 è il primo riferimento della variante. Per una domanda di tipo variante le cose coincidono, mentre per esempio per una domanda di tipo misura nella colonna scelta vi è la misura digitata e nella 1 la misura standard più vicina.

Procediamo nella modifica della regola notando che la voce 03 vuole che l'operatore inserisca manualmente lo spessore.

Per fare questo entriamo nell'editor delle regole e creiamo una nuova variante *"misura"*, come in Figura 3.13:

L	Variante	Tipo	Alias	Validità	Descrizione	Col.Codice	Comando	Va	Minimo	Massimo
☐	MODELLO	Variante								
☐	BASE	Gruppo								
☐	TIPO	Variante								
☐	misura	Misura		$(tipo:1)==03	Misura				10	80
☐										

Figura 3.13: Creazione variante *"MISURA"*.

Abbiamo impostato diversi valori:

- Il tipo variante è stato indicato *"MISURA"*;
- la variante è condizionata alla formula $(tipo:1)==03 (il doppio = sta a significare che il confronto per l'uguaglianza è fatto tra stringhe, per cui per esempio 3 è diverso da 03 o 5+5 è diverso da 10; se si vuole invece un confronto numerico basta scrivere $(tipo:1)=3 o $(tipo:1)=1+2;
- questa variante è limitata in un range da minimo 10 a max 80 tra cui l'operatore può scegliere.

Per far funzionare la cosa dobbiamo ancora modificare la terza colonna della variante *"TIPO"* scrivendo il codice vedi Figura 3.14:

Codice	Descrizione	3_
01	Economico	18
02	Standard	25
03	Speciale	$(misura:0)

Figura 3.14: Collegamento tra la variante *"TIPO"* e la variante *"MISURA"*.

Ciò che si nota è che per il funzionamento delle regole, il contenuto di una variante può a sua volta contenere varianti e/o variabili.

In questo modo scegliendo la risposta 03-Speciale, il comando

Codice 9. Variabile tipo di spessore.

```
#LET spessore=$(tipo:3)
```

Andrà a verificare cosa c'è in $(tipo:3) e trovando $(misura:0) sarà come aver scritto:

Codice 10. Variabile misura.

```
#LET spessore=$(misura:0)
```

Avendo digitato per esempio 35 alla voce misura sarà a sua volta come:

Codice 11. Cambiamento del codice della regola in base ai valori inseriti all'interno della tabella di inserimento delle variabili.

```
#LET spessore=35
```

Di fatto si possono fare cose interessanti con le assegnazioni:

Esempio 8. Passaggio del valore di una variabile come parametro di un'altra (1).

Codice 12. Variabile colonna passata come parametro alla variabile tipo.

```
#LET colonna=3
#LET spessore=$(tipo:$(colonna))
```

oppure:

Esempio 9. Passaggio del valore di una variabile come parametro di un'altra (2).

Codice 13. Nome della variabile con un parametro (indice).

```
#LET indice=1
#LET spessore$(indice)=$(tipo:3)
```

In questo caso viene assegnata alla variabile spessore1 il valore di *$(tipo:3)*.

3.5 Variabili di sistema

Oltre alle variabili definibili all'interno della regola e alle varianti, ***3CadEvolution*** mette sempre a disposizione un certo numero di variabili da poter essere utilizzate nella costruzione della regola:

Variabili di sistema

Variabile	Descrizione
$L	larghezza del box
$A	altezza del box
$P	spessore del box (profondità)
$NL, $NA, $NP	correzioni dell'origine del box: (è possibile utilizzare per le regole le misure $L, $A, $P, ma correggere la dimensione esterna del box, aumentandola o riducendola
$UL, $UA, $UP	correzioni della dimensione finale del box
$PM	corrisponde al verso dell'apertura della porta selezionabile tramite l'interfaccia con l'icona porta
$(aperto), $(latosx), $(latodx), $(latoasx), $(latoadx), $(latobsx), $(latobdx):	sono variabili legate alla interfaccia del programma e in particolare ai check impostati a sinistra: assumono come valori 0 o 1

Tabella 3.1: Variabili di sistema.

Oltre a queste possono essere create via script altre variabili che non sono nella lista delle variabili create da regola e/o di sistema.

Proviamo a utilizzare alcune variabili di sistema in questo modo: vogliamo che i fianchi si trovino all'estremità del box e che la loro posizione vari se si cambia la larghezza del BOX. Inoltre vogliamo che con il check LATOSX e LATODX si attivino o si disattivino i fianchi.

Procediamo intanto con il posizionamento del BOX:

Esempio 10. Spostamento del punto d'origine di un box.

Codice 14. L, sl dipendenti dalle variabili.

```
1 #LET spessore=$(tipo:3)
2 001,L=$(spessore) // Test
3 001,sl=$l-$(spessore),L=$(spessore)
```

Nella terza riga abbiamo messo SL=$l-$(spessore). Abbiamo cioè spostato il pannello in modo che il punto origine sia posizionato con coordinate pari a larghezza del box ($l) – spessore del fianco ($(spessore)). Variando a questo punto la larghezza Figura 3.15, vedremo il secondo fianco spostarsi.

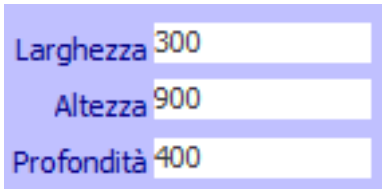


Figura 3.15: Form di variazione altezza, larghezza e profondità.

3.6 Condizioni

Le condizioni sono espresse attraverso il costrutto *#IF* (se): abbiamo scritto che se il valore numerico della variabile $(latosx) è zero allora va inserito tutto quello che c'è nelle righe fino al comando di *#ENDIF*.

N.B: Il costrutto *#IF* è in realtà molto più complesso in quanto è ricorsivo, e può avere condizioni di *#ELSEIF* (altrimenti se) e *#ELSE* (altrimenti).

Facciamo la stessa cosa con la variabile latoDX e proviamo a giocare con l'interfaccia per vedere rispettivamente sparire il fianco a sx o quello a dx o entrambi.



Figura 3.16: Zoccolo dx e sx.

3.7 Tipologie e default

Abbiamo disegnato due oggetti all'interno del box e li abbiamo chiamati fianchi. In realtà abbiamo solo richiamato una tipologia 001 senza preoccuparci di capire né cos'è una tipologia né il suo comportamento.

Affrontiamo questo tema dicendo che una *tipologia* è un raggruppamento di componenti grafici per macro-famiglie, si tratta di un raggruppamento abbastanza arbitrario e non soggetto a vincoli.

Possiamo sviluppare un intero catalogo utilizzando una sola tipologia, ma vi sono delle ragioni pratiche per cui è meglio non farlo.

- Una tipologia definisce un comportamento di default dell'oggetto. Nell'esempio precedente abbiamo impostato solo la formula per la larghezza del pezzo, non quelle per l'altezza o la profondità o il colore o il tipo di disegno (parallelepipedo), questo perché gli altri parametri erano intrinsechi della tipologia;
- se devono essere fatte elaborazioni (esempio stampe escludendo gruppi di componenti dal disegno oppure creazione di programmi per macchine utensili) è molto comodo individuare solo opportune tipologie all'interno del disegno;
- nella regola dovrebbero stare solo le modifiche ad alcuni parametri, mentre per modifiche più generali si dovrebbe variare la tabella delle tipologie in modo tale che se un catalogo diventa anche complesso con molte regole le variazioni possano essere facilmente individuate;
- è più facile lavorare con le tipologie;
- la distinta a formule e listino a formule sono legati a ogni singola tipologia.

E' meglio avere a disposizione delle tipologie *speciali* per gestire informazioni quali vuotature e testi.

Gli oggetti non parametrici come: disegno legato a tabella e zero informazioni legate alla produzione(distinta base), possono stare su una sola tipologia.

Vanno invece distinte le tipologie per gli elementi a riconduzione e lavorazione sugli stessi, visto che questa operazione si basa sulle tipologie.

Vediamo a questo punto lo schema delle tipologie accedendo tramite il menù delle regole.

Posizionando il cursore su una riga contenente una tipologia numerica premiamo il tasto Ctrl+M e ci troveremo nell'interfaccia di gestione della tipologia. Vi sono diversi campi che in alcune situazioni sono definiti , con valori numerici o formule, Figura 3.17 nella pagina 41.

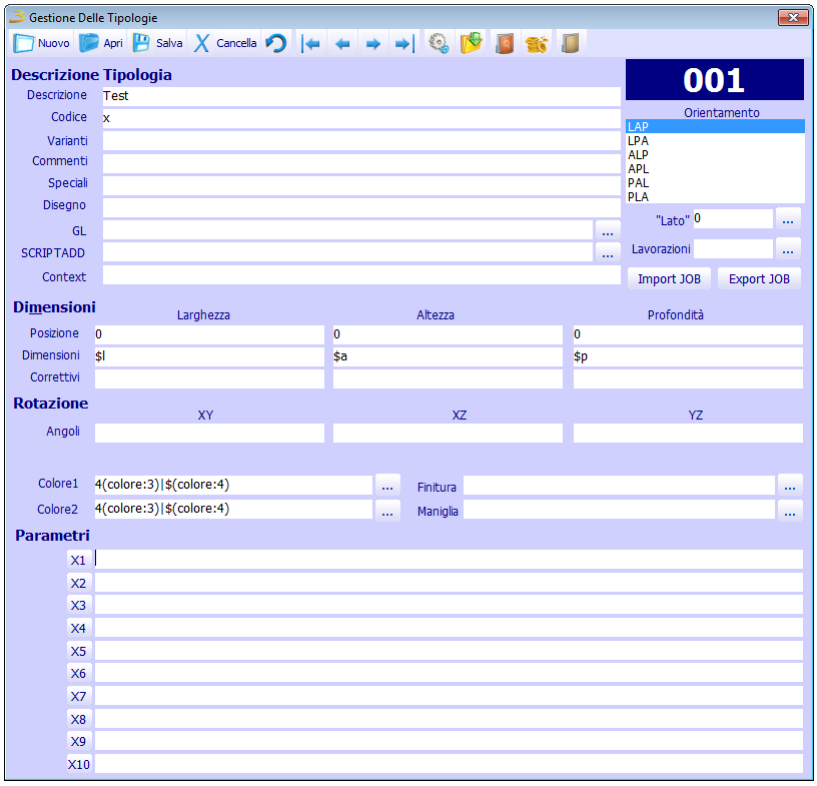


Figura 3.17: Interfaccia di gestione della tipologia.

- In alto a DX della finestra c'è il numero assegnato alla tipologia;
- Descrizione: Serve solo come commento alla tipologia: individua di cosa si sta parlando;
- Codice: se si vuole che la tipologia crei un codice di un articolo da portare a preventivo qui va inserita la formula per il codice; 'X' o '' significano che la tipologia non scarica alcun codice;
- Varianti, Commento, Speciali: completamento per il codice articolo che viene riportato nel preventivo;
- Disegno: Codice del Disegno (sagoma o oggetto 3D) attivato solo in funzione del LATO;
- Lato: forma base di rappresentazione dell'oggetto. Assume i valori riportati a SX. E' importante che questi valori condizionano cosa viene fatto della tipologia;
- Posizione L, A, P: formule per il posizionamento dell'oggetto rispetto al box
- Dimensioni L, A, P: parallelepipedo di ingombro dell'oggetto: anche se la forma non è un parallelepipedo posizione e dimensioni servono a impostare i punti e gli ingombri;

- Correttivi: correzioni gestionali alle dimensioni grafiche: è possibile incrementare o decrementare le dimensioni del pannello solo ai fini di gestione di interfaccia. In molti casi ciò risulta comodo: si pensi al caso delle schiene dei mobili in cui è più comodo lavorare con le dimensioni in luce salvo poi correggerle per avere quella reale;
- Angoli: rotazione del pannello nei tre angoli ;
- Colore1 e Colore2: finitura di base del pannello: il pannello può avere altri riferimenti ai colori tramite le estensioni, ma di norma utilizza 1 al più 2 colori. Il campo colore è un campo multiplo con separatore | in cui viene rappresentato:
  1. Il codice del colore nella lista dei materiali (da 1 a 999).
  2. L'immagine (JPG|TGA|BMP|DRG1) da associare al materiale.
  3. RGB dell'immagine. Da specificare i tre valori divisi da ;.

     Esempio 11. Colore espresso in RGB

     100;200;50
  4. Angolo rotazione immagine per la mappatura.
- Finitura: numero dello script per la ridefinizione del comportamento dell'oggetto (viene gestito solo se lato=0 e ridefinisce il disegno);
- Maniglia: come per la finitura, ma con parametri aggiuntivi per gestire il posizionamento di una eventuale maniglia;
- Parametri x1-x10: sono campi multipli i cui valori sono separati da /. Vengono interpretati dalla distinta base a formule e/o dagli script delle ante. Con i valori LATO<>0, i primi quattro sono utilizzati per il disegno base con significati diversi per ogni LATO.

  I parametri sono utilizzati in questo modo per i vari lati:
  - LATO 9 e 10 (testi): si utilizza il campo commento per il testo da scrivere e il parametro 1 per il tipo di testo, il parametro 2 per la dimensione ;
  - LATO 19 , 20, 25, 26(cornici): si usa il parametro 1 per il percorso della cornice e il parametro 3 e 4 per angolo iniziale e finale;
  - LATO 22(montante angolato): parametro 1 per angolo iniziale in gradi e parametro 2 per angolo finale;
  - Lato 28(luce): parametro 1 per il tipo di luce, parametro 2 e 3 per angolo dello spot.

Siamo pronti a questo punto a creare delle nuove tipologie per la gestione del mobile. Creeremo le seguenti:

- 002 Fianco (e sottotipologia 002.001 fianco DX)
- 003 Ripiano di Struttura
- 004 Ripiano mobile
- 005 Schienale
- 006 Frontale

- 007 interno cassetto
- 008 top a riconduzione
- 010 Elemento grafico generico
- 011 Testo

Procediamo intanto con quelle dalla 2 alla 5, come dalle immagini riportate in Figura 3.18, Figura 3.19, Figura 3.20, Figura 3.21.

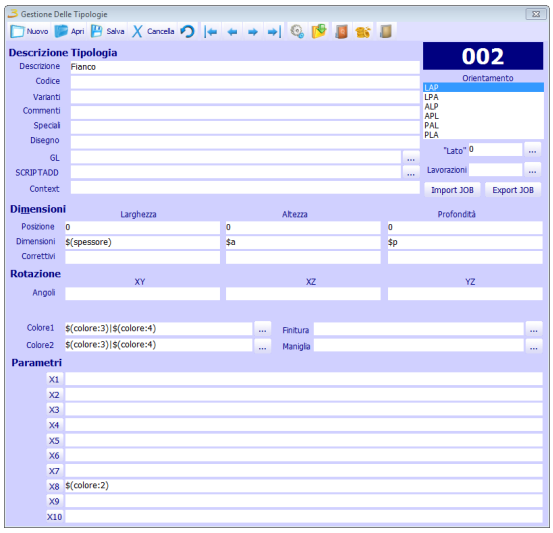


Figura 3.18: Tipologia 002 rappresentante Fianco.

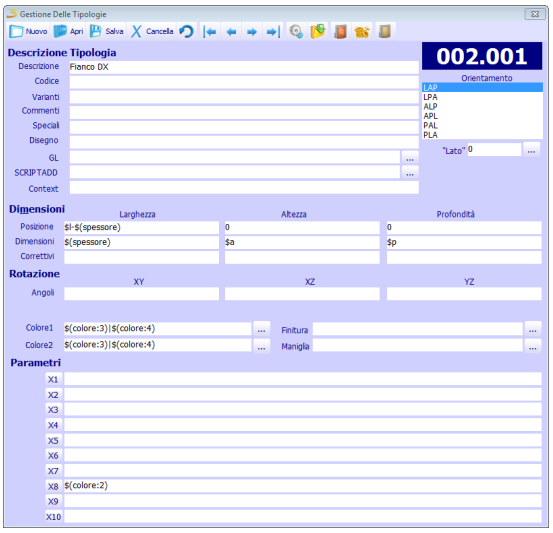


Figura 3.19: Sottotipologia 002.001 rappresentante Fianco DX.

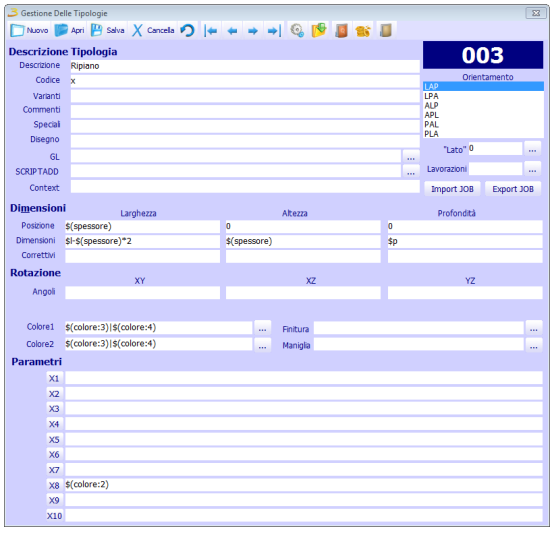


Figura 3.20: Tipologia 003 rappresentante Ripiano.

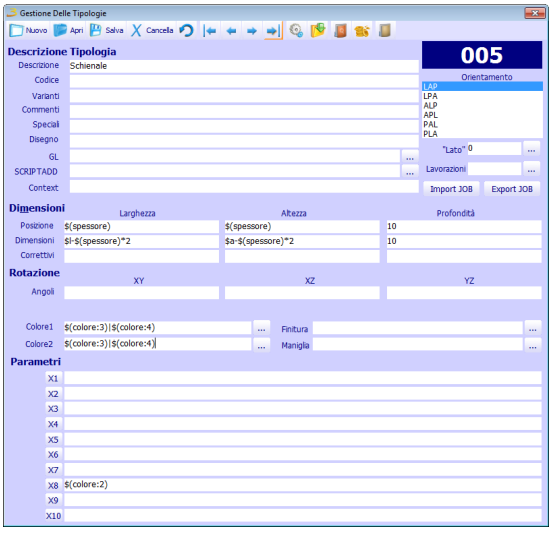


Figura 3.21: Tipologia 005 rappresentante Schienale.

Di seguito senza riportare ogni volta tutta la schermata della tipologia riporteremo esclusivamente lo script per la sua creazione ossia lo stesso che

si ottiene esportando i dati in formato testo (*Setup* applicazione, *Backup* e Esporta BDR dopo aver selezionato tutto (file *MIO* *MIO.BDR*). Parleremo successivamente dei metodi di esportazione e importazione dei dati.

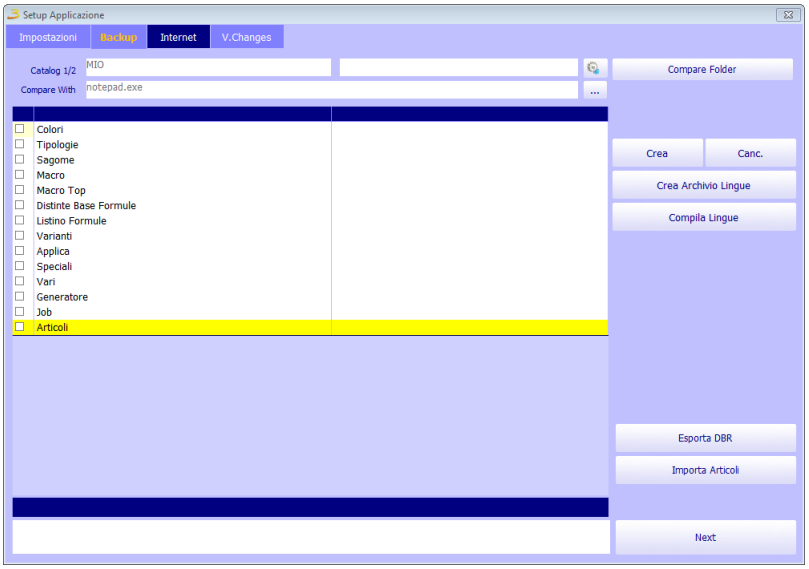


Figura 3.22: Esportazione dati.

L'impostazione delle tipologie seguirà quindi lo schema:

Esempio 12. Script di impostazione delle tipologie.

Codice 15. Tipologie esportate.

```
{ CATALOGO
{ TIPOLOGIE
{ 001 // Test
*AZZERA* =0
DESCRIZIONE =Test
CODICE =x
LATO =0
POSIZIONE =0,0,0
DIMENSIONE =$l,$a,$p
COLORI =$(colbase),$(colbase)
ORIENTAMENTO =LAP
}
{ 002 // Fianco
*AZZERA* =0
DESCRIZIONE =Fianco
LATO =0
POSIZIONE =0,0,0
DIMENSIONE =$(spessore),$a,$p
COLORI =$(colbase),$(colbase)
ORIENTAMENTO =LAP
}
{ 002.001 // Fianco DX
*AZZERA* =0
DESCRIZIONE =Fianco DX
```

```
POSIZIONE =$l-$(spessore)
COLORI =$(colbase),$(colbase)
}
{ 003 // Ripiano
*AZZERA* =0
DESCRIZIONE =Ripiano
CODICE =x
LATO =0
POSIZIONE =$(spessore),0,0
DIMENSIONE =$l-$(spessore)*2,$(spessore),$p
COLORI =$(colbase),$(colbase)
ORIENTAMENTO =LAP
}
{ 004 // Ripiano Interno
*AZZERA* =0
DESCRIZIONE =Ripiano Interno
LATO =0
POSIZIONE =$(spessore),0,20
DIMENSIONE =$l-$(spessore)*2,$(spessore),$p-40
COLORI =$(Colbase),$(Colbase)
ORIENTAMENTO =LAP
}
{ 005 // Schienale
*AZZERA* =0
DESCRIZIONE =Schienale
LATO =0
POSIZIONE =$(spessore),$(spessore),10
DIMENSIONE =$l-$(spessore)*2,$a-$(spessore)*2,10
COLORI =$(colbase),$(colbase)
ORIENTAMENTO =LAP
}
}
}
}
```

A questo punto modifichiamo la regola scrivendola nel seguente modo:

Codice 16. Regola per il mobile.

```
#LET spessore=$(tipo:3)
002 // Fianco
002.001 // Fianco
003 // Ripiano
003,SA=$a-$(spessore) // Ripiano
005 // Schienale
```

Abbiamo disegnato il mobile presente nella Figura 3.23 nella pagina 46 e si tratta di un prodotto variabile sulle dimensioni.

3.8 Dragger

Il *dragger* è una funzione attivabile con una tipologia che permette di inserire in grafica degli oggetti speciali (frecce o pulsanti) utili al dimensionamento, alla modifica delle regole e alla visualizzazione di *PopUp*.

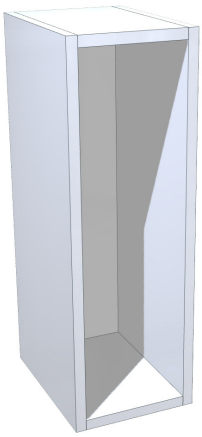


Figura 3.23: Mobile con dimensioni variabili.

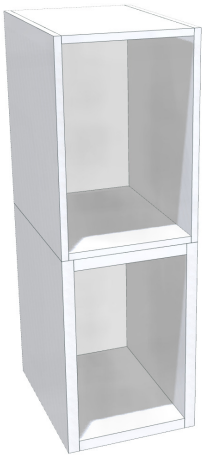


Figura 3.24: Mobile con dimensioni variabili e utilizzo della variante *"RADDOPPIA"*.

Esempio di regola:

Sia 700 una tipologia con tipolato dragger (ovvero il 14). Inseriamo la seguente riga all'interno di un box:

Esempio 13. Regola con utilizzo del della dragger.

Codice 17. Definizione di un dragger.

```
700,SL=$L,SA=$A/2,SP=$P,L=$L,X1=6,X3=trascinami;
L,X2=100;200;300;400;500;600;700 // Dragger
```

Dove:

- SL, SA, SP: sono la posizione del dragger entro il box;
- L: è il valore di partenza del dragger;
- X3: è un campo composto da:
  1. Una descrizione.
  2. Il nome della variabile o della variante a cui si vuole assegnare il valore finale del dragger (può essere anche L, A, P che corrispondono a larghezza, altezza, profondità del box).

Tipi di dragger (X1):

Si presti attenzione a scegliere correttamente il tipo di dragger in base a quello che si vuole fare.

Un dragger può essere:

- Orizzontale o verticale a seconda che le frecce che li rappresentano vengano disegnate o verticalmente rispetto al box o orizzontalmente (scopo puramente visivo);
- alterare o meno la posizione del box. Ad esempio allargare un box verso sinistra significa contemporaneamente spostare il box verso sinistra e aumentarne la dimensione in larghezza della stessa quantità dello spostamento. Vi sono tipologie di dragger che agiscono automaticamente in tale senso e sono quelle indicate con la voce SPOSTAMENTO BOX.

Spostamento box

Descrizione	Pos ID	Sposta box
Dragger orizzontale larghezza sul lato destro	0	No
Dragger orizzontale larghezza sul lato sinistro	1	Verso sinistra
Dragger orizzontale profondità in avanti	2	No
Dragger orizzontale profondità all'indietro	3	All'indietro
Dragger orizzontale interno di profondità	4	No
Dragger orizzontale interno di larghezza	5	No
Dragger verticale larghezza lato destro	6	No
Dragger verticale larghezza lato sinistro	7	No
Dragger verticale interno larghezza	8	No
Dragger verticale interno altezza	9	No
Dragger verticale altezza sopra	10	No
Dragger pulsante	12	No
Dragger pulsante	13	No
Dragger orizzontale larghezza sul lato sinistro	14	No
Dragger orizzontale profondità all'indietro	15	No
Dragger verticale larghezza lato sinistro	16	Verso sinistra
Dragger verticale altezza sotto	17	Verso il basso

Tabella 3.2: Spostamento box.

3.9 Creazione di una macro e riutilizzo del codice

Prendiamo il codice della regola e lo trasformiamo in questo modo:

1. Selezioniamo tutto il codice dalla tipologia 2 alla tipologia 5, copiamo nella clipboard Ctrl+C.
2. Andiamo su una riga vuota e inseriamo il seguente testo:

Esempio 14. Creazione della macro BOX.

Codice 18. Macro BOX.

```
BOX
002 // Fianco
002.001 // Fianco
003 // Ripiano
003,SA=$a-$(spessore) // Ripiano
005 // Schiena
```

3. Premiamo Ctrl+M stando sulla riga 1 (BOX) e incolliamo il testo nella clipboard ottenendo in questo modo di creare una macro BOX con le seguenti righe:

Codice 19. Contenuto della macro BOX.

```
002 // Fianco
002.001 // Fianco
003 // Ripiano
003,SA=$a-$(spessore) // Ripiano
005 // Schienale
```

4. Usciamo dalla macro dopo aver salvato e modifichiamo lo script principale eliminando tutte le tipologie:

Esempio 15. Utilizzo della macro BOX.

Codice 20. Utilizzo della macro BOX.

```
#LET spessore=$(tipo:3)
BOX
```

5. A questo punto nello script principale esistono solo 2 righe, la seconda delle quali richiama le cinque tipologie che servono a disegnare il box.

Proviamo a vedere se le modifiche funzionano fino a questo punto.

Se le cose sono state impostate correttamente il disegno dovrebbe essere rimasto identico, soltanto che lo script di base si è accorciato. Se così non fosse ripetere l'operazione dall'inizio con attenzione.



Figura 3.25: Variante raddoppia.

Dopo questa prima operazione proviamo a modificare la regola creando una variabile *"RADDOPPIA"* di tipo SI/NO, come in Figura 3.25 nella pagina 49.

Entriamo quindi nella grafica e impostiamo le seguenti modifiche, codice 21.

Esempio 16. Utilizzo della variante raddoppia all'interno della regola.

Codice 21. Variante raddoppia.

```
#LET spessore=$(tipo:3)
#IF $(raddoppia:0)
   BOX,a=$a/2
   BOX,sa=$a/2,a=$a/2
#ELSE
   BOX
#ENDIF
```

In questo modo la variabile *"RADDOPPIA"*, prevederà l'utilizzo di due BOX sovrapposti al posto di uno e nelle stesse posizioni, semplicemente modificando i parametri A e SA (altezza e spostamento altezza) come nelle tipologie.

Vediamo che non esiste più l'assegnazione della variabile spessore, ma che questa è stata passata come parametro alla macro. La cosa può comunque creare qualche inconveniente nel caso si dimentichi di farlo. In questo caso rientrando a vedere il contenuto della macro BOX inseriamo sulla prima riga:

Codice 22. Modifiche alla macro BOX necessarie all'utilizzo della variante spessore.

```
!PARAMETRI Spessore=$(tipo:3)
002 // Fianco
002.001 // Fianco
003 // Ripiano
003,SA=$a-$(spessore) // Ripiano
005 // Schienale
```

Questo assicura un valore valido alla macro qualora non ne sia passato alcuno, consigliamo sempre di farlo. La regola principale può diventare:

Esempio 17. Regola rappresentante un box doppio.

Codice 23. Box doppio.

```
ASSEGNA
#IF $(raddoppia:0)
   BOX,A=$a/2
   BOX,SA=$a/2,A=$a/2,SPESSORE=15
#ELSE
   BOX
#ENDIF
```

Ovviamente le macro possono contenere al loro interno il richiamo di altre macro e vedremo in seguito quanto siano utili queste funzioni.

Consigliamo un utilizzo delle macro in ogni situazione possibile, e in particolare consigliamo che ogni regola inizi sempre richiamando una macro di assegnazione (il nostro default è ASSEGNA) nella quale impostare le variabili principali da utilizzare in tutte le regole.

Esempio 18. Macro di assegnazione che imposta le variabili principali.

Codice 24. Macro ASSEGNA.

```
// assegnazioni iniziali di default
#LET nl=0,na=0,np=0,ul=0,ua=0,up=0
// assegnazione delle variabili principali
#LET spessore=$(tipo:3)
```

3.10 Creazione della variante di tipo frontale

Nella cartella `TUTORIAL` del CD di installazione abbiamo inserito alcuni file che sono utili per il proseguimento di questo corso. In particolare abbiamo inserito dei file di immagini per rappresentare alcune Ante e alcuni file di disegno che ci saranno utili in seguito.

Questi file vanno copiati nella cartella `MIO` per proseguire. E' evidente che al posto di questi file possono essere utilizzati file di immagini in formato JPG, TGA, BMP e *3DS* creati con qualsiasi tipo di tecnica.

Potete mescolare i vostri file all'esempio che faremo.

Per proseguire procederemo come segue:

1. Impostiamo e modifichiamo alcuni colori.
2. Modifichiamo la variante *"colore"* e impostiamo le varianti *"_TIPOFIN"* (tipo frontale),*"_FIN"* (finitura frontale) e *"_MAN"* (maniglia).
3. Creiamo la macro per i frontali.
4. Colleghiamo il tutto.

3.10.1 Impostazione e modifica di alcuni materiali

3CadEvolution gestisce una tabella di materiali contenente 999 materiali per catalogo. In questa tabella è possibile inserire e modificare le caratteristiche dei materiali da utilizzare.

La tabella dei materiali non è l'unico punto in cui possono essere definiti i materiali, in quanto sulla regola è possibile cambiarne alcune caratteristiche (colori, *Texture*, dimensioni *Texture*) e lo stesso negli script di disegno che vedremo più avanti.

Qualora si dovessero gestire tanti tipi differenti di materiali è opportuno codificare in questa tabella solo i valori principali e i default e poi utilizzare dei modificatori nelle regole.

Vediamo come sono impostati i materiali: nella creazione del catalogo sono già stati creati 4 materiali:

0 – Materiale di errore:(serve al progettista per identificare durante la scrittura di un catalogo quali materiali non sono ancora stati assegnati;

1 – Bianco;

2 – Acciaio;

3 – Vetro.

La schermata per la definizione di un materiale visibile in Figura 3.26, vediamo come esempio l'impostazione del materiale 10 per i legni:



Figura 3.26: Definizione di un materiale.



Figura 3.27: Materiale legno.

- Codice del materiale.
- Descrizione.
- Riferimenti di base: i materiali di base sono forniti con il programma e permettono di classificare degli effetti specifici.

- Trasparenza.
- Coefficiente di riflessione e impostazione del tipo di riflessione del materiale.
- Impostazione RGB del colore base da tabella colori di *Windows*, colore che verrà poi mescolato con la *Texture*.
- Impostazione della *Texture*.
- Coefficienti di scalatura della *Texture*.
- Angolo di rotazione della *Texture*.
- Impostazione della *Texture* che definisce la profondità del materiale. Questa funzione permette di dare una rappresentazione più realistica ad esempio di un legno con una venatura molto accentuata o in altre situazioni dove il materiale che sto rappresentando presenta una ruvidità.
- Impostazione dell'eventuale *Texture* per il disegno in bianco e nero. Questa *Texture* comparirà nel momento in cui si andranno a togliere i materiali dal menù viste.

Vediamo di impostare un materiale Legno (10) e un materiale Laccato (11) con le caratteristiche riportate nel codice 25.

Codice 25. Schema delle caratteristiche dei materiali.

```
{010 // Legno
*AZZERA*
DESCRIZIONE =Legno
RGB =255,255,255
BIANCONERO =255
TIPOFINITURA =1
TEXTURE =#foto\legni\05.drg1
DIMTEXTURE =1000,1000
ANGOLOTEXTURE =0
METALLICA =0
RIFLESSIONE =2
}

{011 // laccato lucido
*AZZERA*
DESCRIZIONE =laccato lucido
RGB =255,255,255
BIANCONERO =255
TIPOFINITURA =1
TEXTURE =#FOTO\ante\11.drg1
DIMTEXTURE =1000,1531
ANGOLOTEXTURE =0
METALLICA =1.3
RIFLESSIONE =0
}
```

3.10.2 Modifica della variante *"colore"*

Apriamo la regola e modifichiamo la variante *"BASE"* di tipo gruppo.

All'interno di questa variante è definita una variante *"COLORE"*. Apriamo la variante e modifichiamone il contenuto in questo modo, Figura 3.28 nella pagina 53:

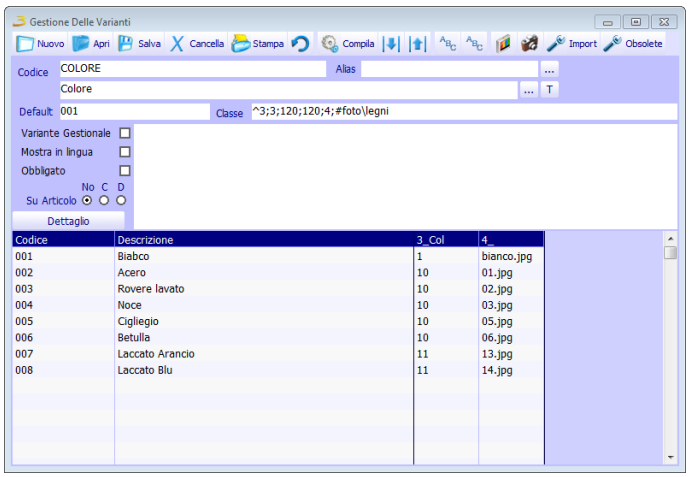


Figura 3.28: Modifica variante colore.

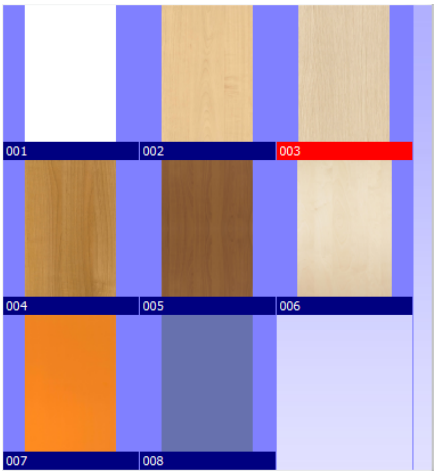


Figura 3.29: Materiali (1).

Il codice è stato inserito con un numero progressivo, ma può essere impostato liberamente utilizzando numeri e lettere (vedremo che i punti permettono di gestire i livelli).

Nella colonna 3 è stato inserito il materiale del colore (1 per il bianco, 10 per i legni e 11 per i laccati).

Nella colonna 4 è stato inserito il codice della finitura.

Nel campo classe vi è un testo il cui significato è legato al tipo di rappresentazione che si vuole dare alla variante in questo caso è un campo separato da [;] che contiene

- Il primo carattere è [^] per indicare la famiglia della classe;
- secondo e terzo sono numero di righe e colonne da visualizzare per la variante;
- quarto e quinto sono la dimensione x e y della *Texture* da rappresentare;
- sesto è la colonna in cui si trova la texture e se anteprima creato con RGB allora aggiungere , e riferimento colonna RGB;
- settimo è il percorso relativo per individuare la *Texture* (ossia #foto legni, dove il carattere # all'inizio individua la cartella in cui è posizionato il catalogo);
- ottavo è il riferimento della colonna per la descrizione;
- nono è il numero di righe per la descrizione.

Memorizzando la variante e uscendo otteniamo che la selezione del colore si presenterà nel seguente modo, Figura 3.31 nella pagina 54.

Questo risultato è connesso al sistema di impostazione delle selezioni per *CLASSE* utilizzando [^]. Esistono comunque altre classi e modalità di visualizzazione, ma questa è sicuramente la più importante per le immagini.

Consigliamo a questo punto di provare a variare le impostazioni della classe per verificare i risultati.

Modifichiamo di nuovo la variante colore per impostare i livelli: ciò che vogliamo ottenere è la separazione tra legni e laccati: per far questo reimpostiamo i codici delle varianti in questo modo, Figura 3.30:

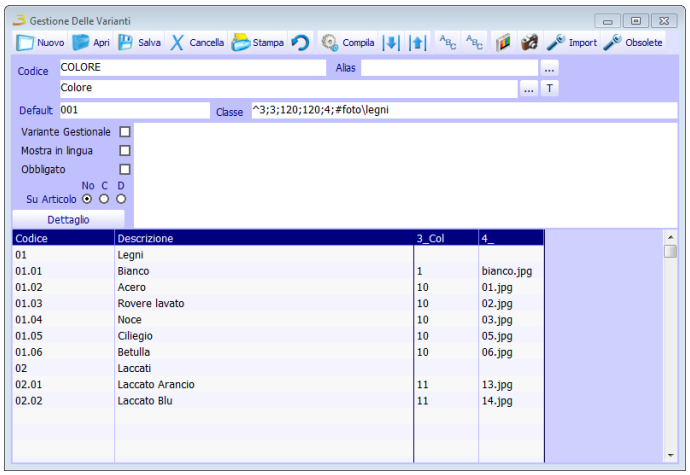


Figura 3.30: Impostazione dei livelli per la variante colore.

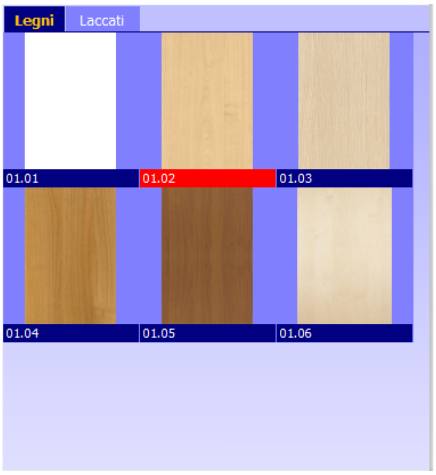


Figura 3.31: Materiali (2).

Notiamo come siano stati inseriti dei codici con [.] . A significare la suddivisione delle varianti. Questo sistema può essere utilizzato a più livelli (massimo due per la classe di visualizzazione [^]) e presenta diverse cose interessanti:

- Le maggiorazioni di prezzo possono essere applicate per livello e quindi non è necessario specificarle per singola varianti;
- le limitazioni possono essere fatte sul genitore;
- l'utilizzatore ha a disposizione un'impostazione logica per le varianti con tante opzioni;
- si possono gestire nella regola meno varianti.

Il tutto ora si presenta come in Figura 3.31.

Fatto ciò, resta da collegare il risultato della selezione alla visualizzazione della variante. Per far questo dobbiamo modificare le varie tipologie fin qui inserite, modificando il campo Colore1 e Colore2, modificando la macro *ASSEGNA*, codice 26.

Esempio 19. Collegamento della variante colore alla visualizzazione.

Codice 26. Macro ASSEGNA con la definizione del colore base.

```
#LET nl=0,na=0,np=0,ul=0,ua=0,up=0
// assegnazione variabili principali
#LET spessore=$(tipo:3)
// colore per la struttura
#LETS colbase=$(colore:3)|$(.percorso)\foto\legni\$(colore:4)
```

> **N.B.:** La modifica delle tipologie permette di reimpostare caratteristiche su tutte le regole fin qui definite, ma in alcuni casi è necessario ricompilarle perché queste caratteristiche abbiano effetto.

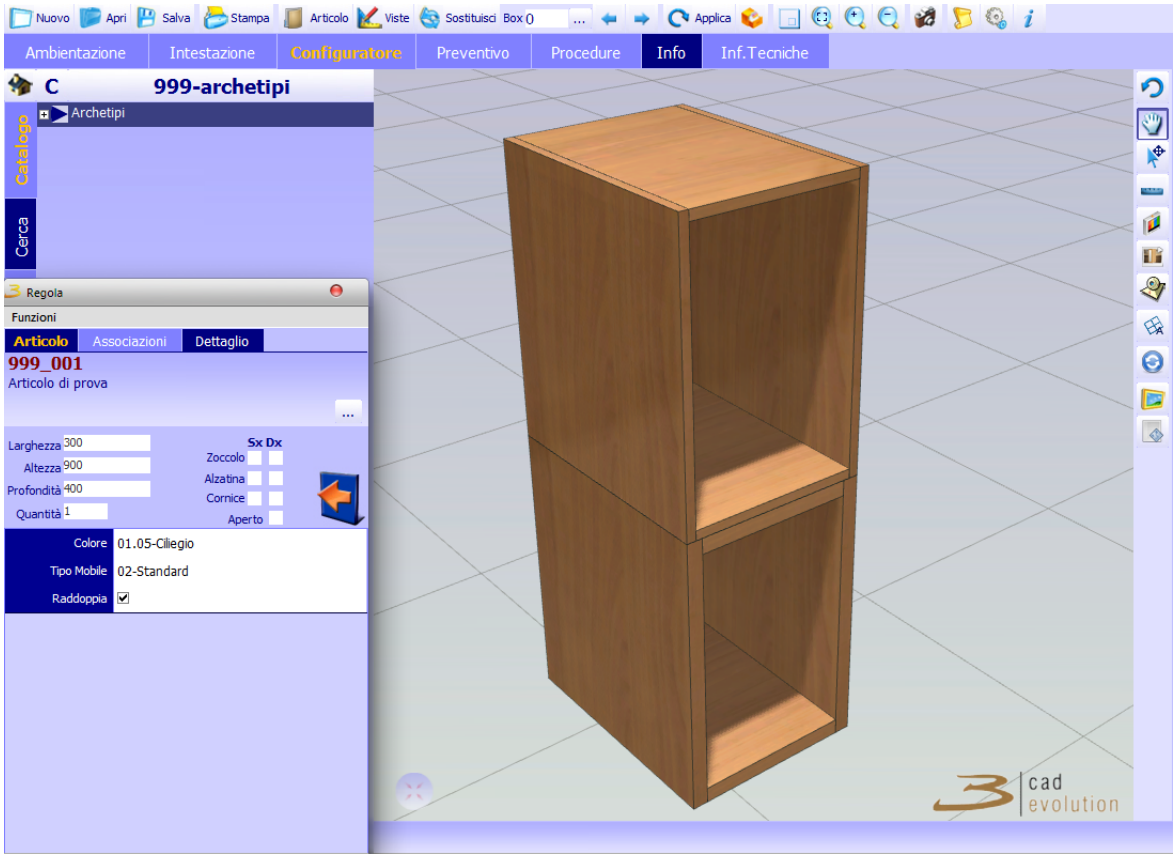


Figura 3.32: Mobile con il materiale legno applicato.

Vediamo che per assegnare la variabile *COLBASE* abbiamo utilizzato il comando *#LETS* (assegnazione di un valore stringa) e non il numerico; abbiamo utilizzato anche la variabile di sistema $(.percorso) anziché il nome del catalogo per individuare la posizione della *Texture* e non abbiamo impostato un riferimento assoluto. Evitare di scrivere nelle regole riferimenti assoluti a file del tipo:

- `C:/EVOLUTION/ECADPRO/MIO/FOTO/LEGNI...;`
- `IO/FOTO/LEGNI....`

3.10.3 Impostazione delle altre varianti

Impostiamo una variante *"tipo finitura"* che determina il tipo di finitura per il frontale del mobile che stiamo costruendo, con le seguenti opzioni:

- A Giorno con ripiani legno;
- a Giorno con ripiani vetro;
- con Ante;
- con Ante vetro;
- con Cassetti.

Costruiamo la variante come segue e colleghiamola alla regola, come abbiamo fatto precedentemente con le altre. Costruiamo quindi una variante per i colori dei frontali (ha lo stesso tipo di informazioni della variante colore) e una variante per la gestione delle maniglie: quest'ultima, oltre al disegno rappresentativo imposta una serie di caratteristiche utili al disegno 3D che relazioneremo di seguito con uno script di disegno.

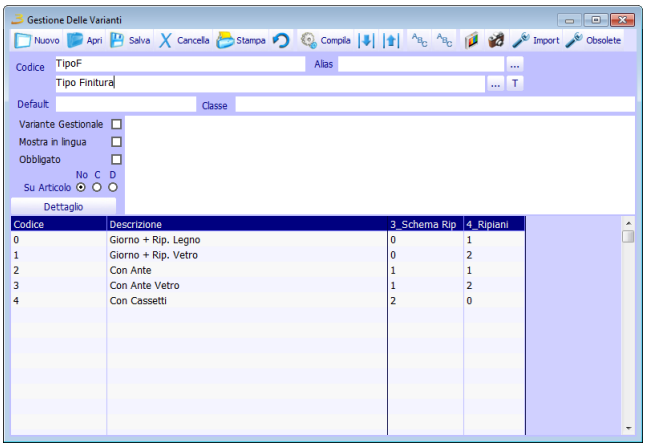


Figura 3.33: Modifica variante Tipo Finitura.

A questo punto definiamo una tipologia per i frontali, ante e cassetti. Questa tipologia utilizzerà una cosa in più rispetto alle precedenti e cioè una finitura. Questo si rende necessario per facilitare il lavoro di disegno e per impostare alcune caratteristiche chiave come la possibilità di aprire e chiudere le ante e posizionare in modo automatico le maniglie.

Le finiture operano attraverso una interfaccia di tipo vbscript, ossia utilizzano un file *ASCII* che contiene codice VBSCRIPT che legge alcune informazioni dalla tipologia da disegnare e in base a queste richiama funzioni nell'engine grafico che provvedono al disegno.

La scrittura di una finitura è un argomento di per se molto complesso che richiede un certo livello di conoscenza del linguaggio visual basic e delle tecniche di programmazione legate al *CAD*. Alla fine del manuale vi è un riferimento alle funzioni messe a disposizione del motore grafico, ma comunque

il consiglio è quello di utilizzare questi script con una certa parsimonia e soprattutto riutilizzare quelli esistenti, magari presi da altri cataloghi, come faremo di seguito con ante e maniglie.

Le finiture di un catalogo stanno all'interno del file *_ECAD_ante.vbs* e sono funzioni dichiarate nella forma:
Sub AntaXX(dl,da,dp,c1,c2,pTerra,lato,px,py,pa,pAltezza,pLarghezza)

oppure:
Sub ManigliaXX(dl,da,dp,c1,c2,pTerra,lato,px,py,pa,pAltezza,pLarghezza)

Dove XX sta per un numero intero che definisce nel primo caso la finitura e nel secondo il gruppo di finitura (il numero assegnato all'interno della tipologia per la maniglia è composto da 10x(numeromaniglia)+(latoapplicazione).

Senza entrare in dettaglio, nel catalogo MIO sono state copiate le ante 1 (ANTA LEGNO), 2 (ANTA VETRO), 50 (LIBRERIA) e la maniglia.

Utilizzeremo pertanto questo formato per la rappresentazione delle ante.

Per poter procedere quindi definiamo una tipologia 11 Anta e 12 Anta Vetro, 13 cassetto e 14 interno cassetto, più una variante *"_fin"* che rappresenta il tipo di finitura. Tutte queste sono collegate al sistema utilizzato nei cataloghi generici forniti con il programma ("CUCINA", "DAUBAGNO", "DAUWALL" e "PARETI") e per quanto sia possibile riprogettare il tutto in un modo completamente diverso ci troviamo qui gran parte del lavoro fatto.

Nella cartella `INS01` all'interno del percorso _ecadpro/doc troviamo il file *ANTE.TXT* (parliamo successivamente della strutturazione di questo file che serve per l'importazione e esportazione dei dati di ***3CadEvolution*** e importiamo il suo contenuto, come riportato nella Figura 3.34 nella pagina 58 accedendo al menu *Setup*.
Nella Figura 3.35 nella pagina 58 possiamo vedere la nuova impostazione delle tipologie fino a questo momento. Notiamo in particolare che per le ante sono state definite anche delle sottotipologie, questo per facilitare le cose in fase di impostazione delle regole.

Entrando nelle tipologie notiamo che le sottotipologie ridefiniscono semplicemente alcune informazioni, tra di loro collegate. Diventa più semplice quindi effettuare la manutenzione su una sottotipologia, quando il cambiamento (come nel caso delle ante) riguarda una serie di campi.

In particolare entriamo nella tipologia 011 e osserviamo con attenzione come sono stati definiti i vari campi:

I valori in questione sono indicativi, in quanto lo script *manigliaxx* potrebbe cambiarne in tutto o in parte modalità e funzionamento.

Quindi il numero 13 significa maniglia 1 con orientamento destra centro.

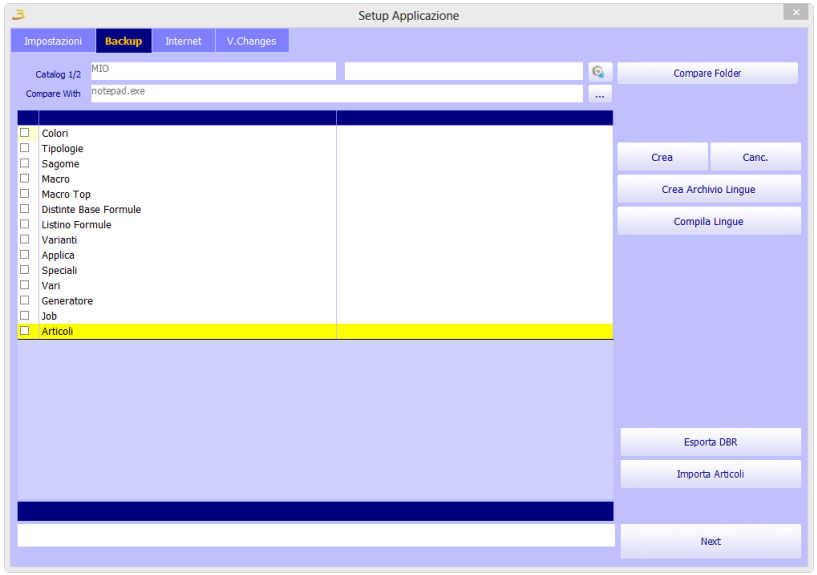


Figura 3.34: Menu setup, importazione e esportazione dei dati.

Codice	Opzione
001	Test
002	Fianco
002.001	Fianco DX
003	Ripiano
005	Schienale
011	Anta SX
011.001	Anta DX
011.002	Anta Apertura Vasistas
011.003	Anta Ribalta
011.010	Fascia di chiusura
012	ANTA Vetro (SX)
012.001	ANTA Vetro (DX)
012.002	ANTA Vetro Vasistas
012.003	ANTA Vetro ribalta
013	Cassetto e Cestone
014	INTERNO CASSETTI - CESTONI

Figura 3.35: Lista delle tipologie.

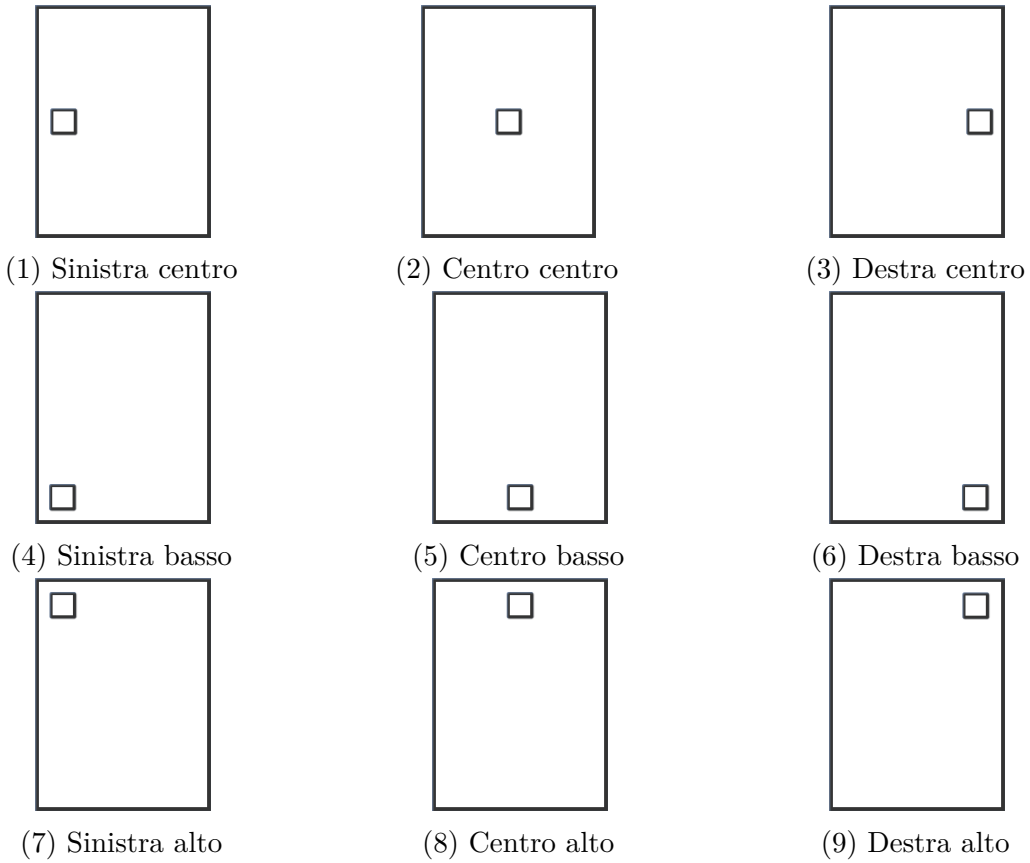


Figura 3.36: Posizione maniglia.

Adesso vediamo una serie di dati impostati nei parametri aggiuntivi, Figura 3.37 nella pagina 59.

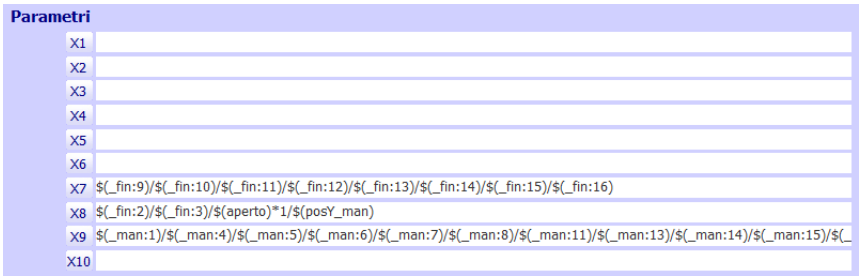


Figura 3.37: Parametri aggiuntivi.

Questi dati sono utilizzati dagli script e dall'elaborazione della distinta base a formule e prezzo a formule (vedi successivamente). In dettaglio possiamo passare alla tipologia una quantità più alta di informazioni rispetto ai campi base già predisposti. Saranno poi i vari script a utilizzare queste informazioni e a collegarle a un risultato.

Convenzionalmente i parametri 1, 2, 3 sono utilizzati dalla grafica di base, i parametri 6, 7, 8, 9 dalla distinta base a formule e dagli script personalizzati. Per impostare più informazioni all'interno di un parametro aggiuntivo (sotto campi) si usa il carattere `/` come separatore.

Si utilizza questo metodo anche perché non possono essere utilizzate direttamente le variabili della regola, soprattutto nella distinta base, in quanto l'elaborazione avviene per stadi successivi: la distinta base e gli script vedono soltanto i campi contenuti all'interno della tipologia, che sono già stati valutati dalle regole.

Uno degli errori più frequenti nella scrittura del catalogo è proprio relativo al fatto di non tener conto della separazione degli ambienti:

- Le regole sono valutate in una fase che permette di identificare una distinta base di tipologie grafiche;
- l'analisi del contenuto della distinta base delle tipologie grafiche è utilizzato per creare il disegno (tramite i core dell'ambiente grafico e gli script in *_epro_ante.vbs*), per creare la distinta prezzi, la distinta base a formule, le elaborazioni finali (ricalcolo top) e le lavorazioni per il job.

File da copiare

Copiamo inoltre il contenuto della cartella `INS01` all'interno della cartella `MIO`, facendo molta attenzione all'ordine delle cartelle: all'interno della cartella `INS01` si trovano le foto delle ante, e i file di disegno per le maniglie in formato *3DS*.

Le *Bitmap* utilizzate sono state create con programmi di disegno come ***ACDSEE*** o ***Photoshop***, quest'ultimo in particolare per le immagini con trasparenza (TGA) dotate del canale Alpha.

Per i *3DS* o file X ognuno può utilizzare il tool che preferisce (***RYNO***, ***Sketch Up***, ***3DMAX***, ***Blender***).

3.10.4 Impostazione interferenza

Impostiamo una tipologia che utilizzeremo come *interferenza*, durante gli spostamenti.

Vale la pena fare una premessa: nessun oggetto si relaziona in modo automatico con altri oggetti, quindi due o più elementi possono compenetrarsi durante le fasi di posizionamento. Per evitare questo definiamo dei particolari oggetti all'interno della regola di tipo interferenza che bloccano gli spostamenti indesiderati; in realtà possono bloccare anche aree diverse rispetto a quanto disegnato, pensiamo al fatto che se disegniamo una cassettiera, dobbiamo evitare che siano posizionati oggetti in fronte ad essa per consentire l'apertura dei cassetti o altre situazioni analoghe.

Le interferenze sono definite da parallelepipedi (niente oggetti sagomati o particolari) e derivano da tipologie nelle quali è impostato il lato 23.

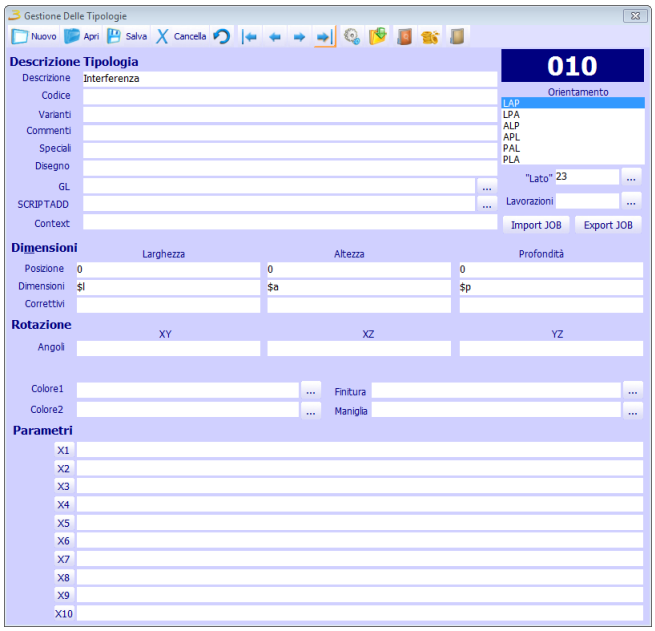


Figura 3.38: Impostazione interferenza.

Entriamo a questo punto nella regola e inseriamo la tipologia 10, con posizione 0,0,0 e dimensioni uguali a quelle del box occupato. Se disegniamo 2 o più componenti vedremo che vengono gestite le collisioni tra oggetti e il posizionamento di più oggetti risulta in questo modo semplificato:

Esempio 20. Posizionamento degli oggetti.

Codice 27. Utilizzo di interferenza.

```
ASSEGNA
#IF $(raddoppia:0)
   BOX,a=$a/2
   BOX,sa=$a/2,a=$a/2,Spessore=15
#ELSE
   BOX
#ENDIF
010 // Interferenza
```

Modifichiamo la regola

Interveniamo a questo punto sulla regola per inserire le nuove varianti che abbiamo definito, e nello stesso tempo vogliamo rimuovere le prove precedenti relative al BOX.

Ci troveremo con la situazione mostrata in Figura 3.39 nella pagina 62 dove abbiamo rimosso la variante *"RADDOPPIA"*, e abbiamo inserito una variante di tipo *"separatore"* che fa apparire una nuova linguetta per motivi prettamente estetici (quando le varianti diventano tante è opportuno utilizzare le linguette come separatori per semplificare la ricerca).

L	Variante	Tipo	Alias	Validità	Descrizione
☐	MODELLO	Variante			
☐	Tipo	Variante			
☐	Colore	Variante			
☐	Misura	Misura		$(tipo:1)==03	
☐	vTop	SiNo			Vuoi Top?
☐	FINITURA	Separatore			
☐	TipoF	Variante			
☐	_Fin	Variante		$(TipoF:3)>0	
☐	_Man	Variante		$(TipoF:3)>0	
☐	Passi	Libero		$(TipoF:3)<2	
☐					

Figura 3.39: Variante di tipo *"separatore"*.

Abbiamo inserito la variante *"TipoF"* (tipo finitura) e condizionata ai valori di questa la variante *"_Fin"* e *"_Man"*, inoltre la variante Passi per il posizionamento dei ripiani.

Come prima cosa inseriamo una sottotipologia 004.001 ripiano vetro (vogliamo sostanzialmente ereditare il ripiano vetro dal ripiano legno, cambiano il contenuto di alcuni campi come nella Figura 3.40:

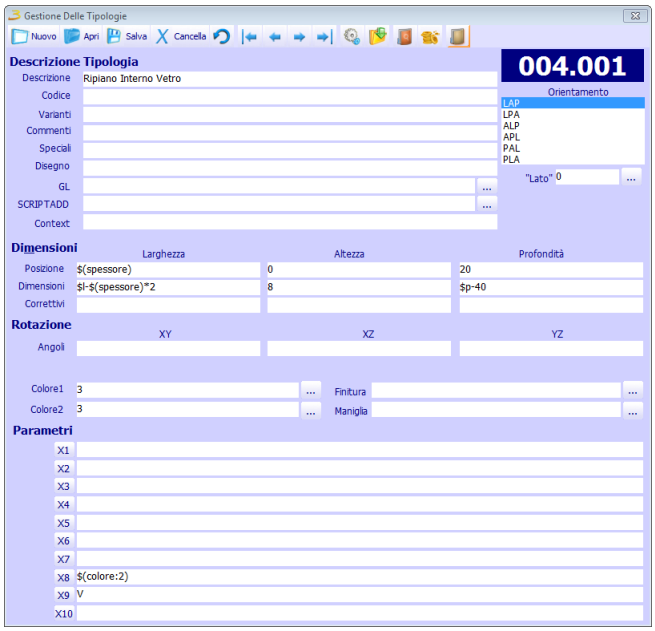


Figura 3.40: Sottotipologia 004.001, ripiano vetro.

- Descrizione ripiano interno vetro;
- altezza 8mm fissa;
- colore 1 e Colore 2 =3 (il materiale che abbiamo definito come vetro);
- parametro 9 = 'V', ci servirà successivamente per capire che scaricheremo un ripiano vetro anziché uno legno.

Passiamo quindi a modificare la macro box nel seguente modo:

Esempio 21. Macro BOX complessa.

Codice 28. Macro BOX.

```
!PARAMETRI Spessore=$(tipo:3),ripiani=0

002 // Fianco
002.001 // Fianco DX
003 // Ripiano
003,SA=$a-$(spessore) // Ripiano
005 // Schienale

#IF $(ripiani)=1
  #FOR $(passi),$(i)>$a-100
    004,sa=$(i)
  #ENDFOR
#ELSEIF $(ripiani)=2
  #FOR $(passi),$(i)>$a-100
    004.001,sa=$(i)
  #ENDFOR
#ENDIF
```

Abbiamo inserito un secondo parametro (ripiani), cui passiamo un valore 0 (senza ripiani), 1 con ripiani legno e 2 con ripiani vetro. Non è obbligatorio dichiarare la variabile nei parametri ma la rende locale e facilita la leggibilità del codice; la variabile *passi*, più avanti non è dichiarata.

Abbiamo messo una condizione per il caso con ripiani legno e per i ripiani vetro.

Esempio 22. Condizione del tipo di materiale per i ripiani.

```
#IF $(ripiani)=1
..
#ELSEIF $(ripiani)=2
..
#ENDIF
```

All'interno abbiamo utilizzato il ciclo for che funziona in questo modo:

- *#FOR* (insieme di valori separati da [;]),(valore di eccezione),(variabile da riempire per default la variabile è $(i));
- la chiusura del ciclo è il comando *#ENDFOR*.

Esiste anche il costrutto *#FOR* $nn:mm (dove nn e mm sono rispettivamente 2 interi: esempio #FOR $1:5 corrisponde a #FOR 1;2;3;4;5.

Abbiamo utilizzato il ciclo *FOR* per posizionare i ripiani alle altezze definite dall'operatore sulla variante libera *"PASSI"*. E' in realtà l'operatore a scrivere le posizioni dei ripiani separate da [;] (questa è una impostazione cablata nel programma), ad esempio:

320;640;960;1280;1600;1920

Per avere tutti i ripiani a passo 320.

Da notare la condizione di non validità all'interno del *FOR*:

Esempio 23. Condizione del comando for.

```
$(i)>$a-100
```

Che serve a dire di non considerare il ripiano qualora la sua posizione in altezza superi quella del box – 100mm.

Modifichiamo anche la regola per permettere di specificare il parametro del tipo di ripiano che dipende dalla colonna 4 della variante *"TipoF"*.

Esempio 24. Tipo di ripiano.

Codice 29. Variante *"TipoF"*.

```
ASSEGNA
BOX,ripiani=$(tipof:4)
010 // Interferenza
```

A questo punto siamo a posto con la parte interna, procediamo con le ante e i frontali.

Abbiamo importato anche 2 macro _Ante e _AnteV, precedentemente, andiamo a analizzare la macro _ante :

Codice 30. Macro ante.

```
// GESTIONE COPPIA ANTE
#IF $L>600 || $(2ANTE)=1
  011,SL=0,L=$L/2-1
  011.1,SL=$L/2,L=$L/2
#ELSE
  #IF $(PMan)=3
    011
  #ELSE
    011.1
  #ENDIF
#ENDIF
```

Questa è una macro molto semplice da interpretare e imposta le ante con un unico inserimento, tenendo conto che oltre una certa larghezza servono 2 ante (oppure se viene forzata la variante *"2ANTE"* a 1) altrimenti posiziona un'anta sinistra o destra in funzione del contenuto della variabile PMAN.

Utilizzeremo ora questa macro per integrare al BOX il seguente codice:

Esempio 25. Box complesso.

Codice 31. Integrazione delle ante.

```
!PARAMETRI Spessore=$(tipo:3),ripiani=0

002 // Fianco
#IF $(tipof:1)=2
  _ante,sl=$(spessore),sa=$(spessore),l=$l-$(spessore)*2,a=$a
     -$(spessore)*2,pman=$pm
#ELSEIF $(tipof:1)=3
  _anteV,sl=$(spessore),sa=$(spessore),l=$l-$(spessore)*2,a=
     $a-$(spessore)*2,pman=$pm
#ELSEIF $(tipof:1)=4
  // Cassetti
#ENDIF
```

Vediamo come abbiamo richiamato le macro _Ante e _AnteV passando nuove posizioni e dimensioni: questo ci permette, di posizionare le ante nel punto giusto (utilizzeremo in seguito queste possibilità con i sistemi di progettazione con aggiunte e divisioni).

Come ultima cosa integriamo le cassettiere, anche qui utilizziamo una macro _Cassetto, nel modo sotto descritto: come prima cosa calcoliamo un numero ideale di cassetti tale da riempire il vano e quindi impostiamo l'altezza reale del cassetto in modo da non avere resti. Questa è la macro _Cassetti:

Esempio 26. Macro cassetti.

Codice 32. Macro _Cassetti.

```
#LET nCass=int($a/160+.5)
#LET hCass=$a/$(nCass)
#FOR $1:$(nCass)
  _cassetto,sa=($(i)-1)*$(hCass),a=$(hCass)
#ENDFOR
```

E quindi all'interno del box scriviamo:

Codice 33. Gruppo della macro cassetti all'interno del box.

```
#ELSEIF $(tipof:1)=4
  // Cassetti
  _Cassetti,sl=$(spessore),sa=$(spessore),l=$l-$(spessore)*2,
      a=$a-$(spessore)*2
#ENDIF
```

A questo punto non ci resta che fare qualche prova di disegno, per ottenere qualche composizione.

3.10.5 Tipologie a riconduzione

Vogliamo inserire un Top sopra il nostro mobile come opzione aggiuntiva, tale elemento decorativo dovrà fondersi in modo automatico con altri elementi per vicinanza.

La gestione dei top, e degli elementi a riconduzione prevede diverse funzioni che saranno riportate in dettaglio nella seconda parte del manuale tecnico, con opportuni esempi, per il momento operiamo come segue.

Creiamo la tipologia 20 con le caratteristiche come in Figura 3.41.

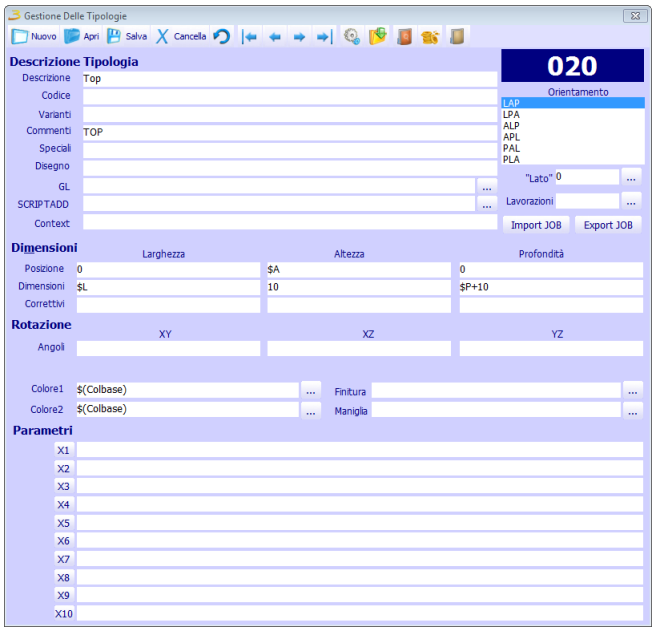


Figura 3.41: Tipologia a riconduzione 020, gestione del top.

Entriamo nella maschera *Speciali* e inseriamo la tipologia 20 come tipologia di tipo Top, come riportato in Figura 3.42 nella pagina 67.

La funzione viene selezionata nel menu speciali alla voce *gruppi di tipologie*.

Fatto questo modifichiamo la regola per inserire una variante di tipo si/no *"vTop"* (vuoi top) e modifichiamo il codice della regola di base, codice 34.

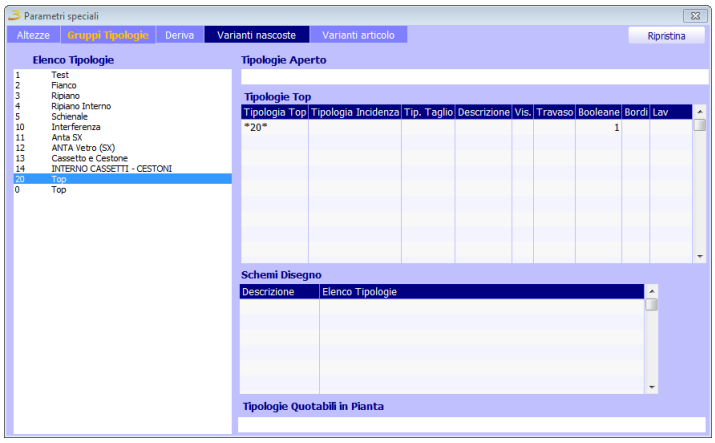


Figura 3.42: Paramentri speciali.

Esempio 27. Variante *"vTop"* di tipo sì/no.

Codice 34. Modifiche al BOX con l'aggiunta della variante *"vTop"*.

```
ASSEGNA
BOX,ripiani=$(tipof:4)
010 // Interferenza
#IF $(vTop:1)=1
   020 // Top
#ENDIF
```

Ci accorgeremo, affiancando 2 box uguali che i loro Top si uniscono a preventivo in un unico elemento, il cui codice è quello che abbiamo scritto nella voce *commenti* della tipologia.

Tale tecnica è molto più estesa di quanto riportato in questa parte del manuale, e consigliamo di approfondirla o analizzando i cataloghi generici di bagni e cucine oppure nella seconda parte del manuale.

3.10.6 Raggruppamento delle variabili in un gruppo

La dichiarazione delle variabili all'interno della regola è stata fin qui sequenziale e abbastanza casuale, abbiamo cioè inserito i dati mano a mano che ci servivano, senza una corretta analisi iniziale. Quando si costruiscono molte regole occorre tuttavia, per evitare di dimenticare dichiarazioni, raggrupparle in gruppi.

In questo modo la modifica a un gruppo di definizione sarà immediatamente disponibile su tutte le regole che lo contengono.

In Figura 3.43 abbiamo la situazione iniziale:

L	Variante	Tipo	Alias	Validità	Descrizione
☐	MODELLO	Variante			
☐	Tipo	Variante			
☐	Colore	Variante			
☐	Misura	Misura		$(tipo:1)==03	
☐	vTop	SiNo			Vuoi Top?
☐	FINITURA	Separatore			
☐	TipoF	Variante			
☐	_Fin	Variante		$(TipoF:3)>0	
☐	_Man	Variante		$(TipoF:3)>0	
☐	Passi	Libero		$(TipoF:3)<2	
☐					

Figura 3.43: Varianti prima del raggruppamento.

Copiamo le varianti da Finitura in giù nella ClipBoard e cancelliamole inserendo al loro posto la riga Finitura di tipo GRUPPO.

A questo punto premendo varianti si apre l'editor dei gruppi, Figura 3.44. Creiamo un nuovo gruppo *Finitura* e incolliamo i dati come in Figura 3.45.

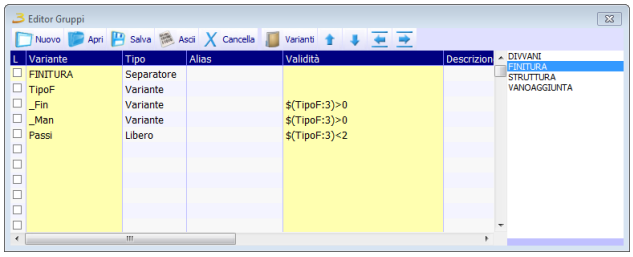


Figura 3.44: Editor gruppi, variante FINITURA.

L	Variante	Tipo	Alias	Validità
☐	MODELLO	Variante		
☐	Tipo	Variante		
☐	Misura	Misura	$(tipo:1)=03	
☐	vTop	SiNo		
☐	FINITURA	Gruppo		
☐				
☐				
☐				

Figura 3.45: Variante FINITURA di tipo gruppo.

In realtà con questa operazione non è cambiato nulla, ma si ha ora la possibilità di utilizzare nella dichiarazione la Finitura come GRUPPO senza dover specificare altri parametri.

Facciamo la stessa operazione per le prime quattro variabili, definendo il gruppo Struttura, Figura 3.46 nella pagina 69, Figura 3.47 nella pagina 69.

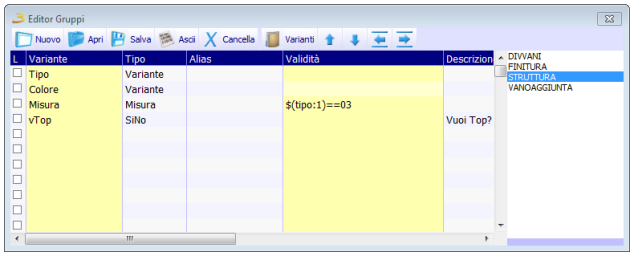


Figura 3.46: Editor gruppi, variante STRUTTURA.

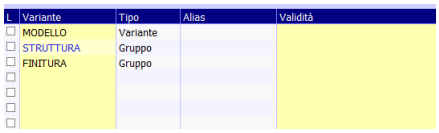


Figura 3.47: Variante STRUTTURA di tipo gruppo.

Procederemo a questo punto con la creazione di una nuova regola da utilizzare per un Box Sagomato.

Creiamo una nuova regola dal menu Regole con nome _002 (menu Gestione Regole), Figura 3.48.

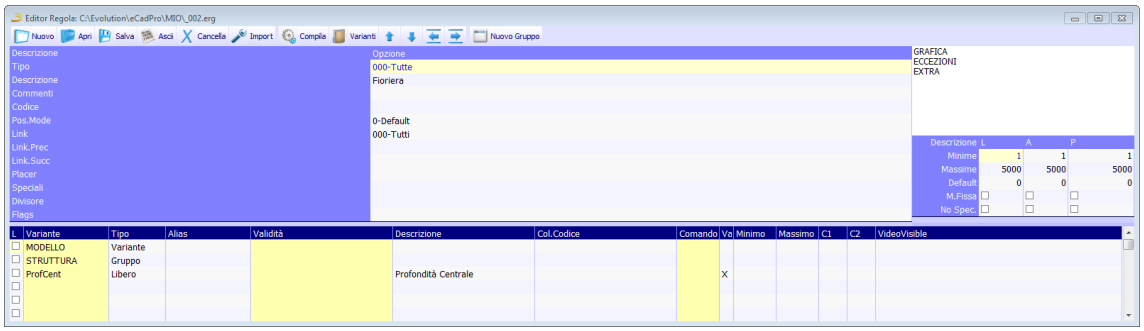


Figura 3.48: Editor regola per un box sagomato.

Creiamo inoltre un articolo da associare a questa regola: basta prendere un articolo esistente, e associargli la nuova regola come nella Figura 3.49 nella pagina 70 e premere il pulsante salva.

Se carichiamo l'articolo Fioriera, non risulterà nulla, quindi dovremo modificare la regola in modo da inserire alcuni elementi grafici.

Procediamo come segue: entriamo nella macro BOX e copiamo le informazioni per incollarle direttamente nella regola (non vogliamo modificare il BOX standard).

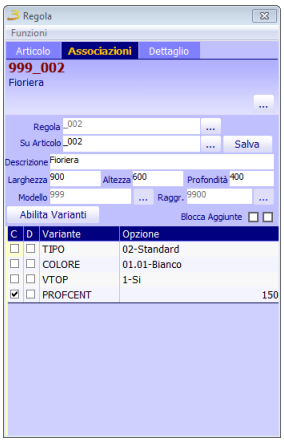


Figura 3.49: Associare regola ad un articolo.

Codice 35. Regola complessa.

```
ASSEGNA
002 // Fianco
002.001 // Fianco DX
003 // Ripiano
003,SA=$a-$(spessore) // Ripiano
005 // Schienale
#FOR 320;640;960;1280;1600;1920,$(i)>$a-100
  004,sa=$(i)
#ENDFOR
```

A questo punto abbiamo disegnato l'elemento standard, ma vogliamo sagomare i ripiani con una forma parametrica, tipo quella di seguito ottenuta con il CAD2D, Figura 3.50:

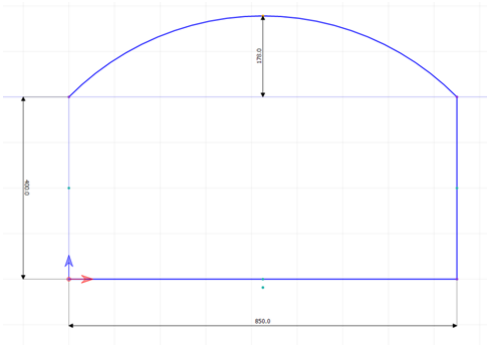


Figura 3.50: Disegno in CAD2D.

Possiamo utilizzare questa sagoma impostandone il codice (nome file), ma purtroppo questa sagoma non si modificherebbe cambiando le dimensioni del BOX, possiamo quindi intervenire impostando un percorso in modo procedurale, la sagoma qui sopra si scriverebbe:

Esempio 28. Utilizzo sagoma.

Codice 36. Inserimento sagoma.

```
!SAGOMA tmp 0,0,850,0,A30,850,400,850/2,400+178,0,400
003,dis=$(tmp) // Ripiano
```

La sagoma viene descritta a punti, separati dal separatore `,` e memorizzata nella variabile TMP impostata poi nel campo DIS quando si deve memorizzare la tipologia.

I punti vanno a coppie X,Y (quindi i primi 2 0,0 e 850,0) sono chiari. Successivamente però vi è un comando A30 che dichiara che i tre punti successivi sono parte di un arco diviso in questo caso in 30 Segmenti. L'arco passa per il punto 850,400 poi per 850/2, 400+178, e quindi arriva al punto 0,400.

Non è necessario chiudere la sagoma in quanto si tratta di un percorso chiuso.

Vediamo che è possibile definire i punti anche con formule e successivamente con variabili. Infatti al posto di 850 scriviamo $l-$(spessore)2 che è la reale larghezza dell'elemento, al posto di 400 $p per i ripiani strutturali e $p-40 per quelli interni e come ampiezza della corda la variabile $(profcent:0), per ottenere la sagoma parametrica.

Vediamo la regola completa che definisce 3 sagome (ripiani strutturali, interni e TOP).

Esempio 29. Regola completa che definisce 3 sagome.

Codice 37. Regola completa di un mobile con utilizzo delle sagome.

```
ASSEGNA
002 // Fianco
002.001 // Fianco DX
!SAGOMA tmp 0,0,$l-$(spessore)*2,0, \
      A30, \
      $l-$(spessore)*2,$p, \
      ($l-$(spessore)*2)/2,$p+$(profcent:0), \
      0,$p
003,dis=$(tmp) // Ripiano
003,SA=$a-$(spessore),dis=$(tmp) // Ripiano
005 // Schienale

!SAGOMA tmp 0,0,$l-$(spessore)*2,0, \
      A30, \
      $l-$(spessore)*2,$p-40, \
      ($l-$(spessore)*2)/2,$p-40+$(profcent:0), \
      0,$p-40

#FOR 320;640;960;1280;1600;1920,$(i)>$a-100
  004,sa=$(i),dis=$(tmp)
```

```
21
22 #ENDFOR
23
24 !SAGOMA tmp 0,0,$l,0, \
25       A30, \
26       $l,$p+10, \
27       $l/2,$p+10+$(profcent:0), \
28       0,$p+10
29
30 #IF $(vTop:1)=1
31   020,dis=$(tmp) // Top
32 #ENDIF
```

Risultato è visibile in Figura 3.51.

Con le sagome possiamo eseguire una infinità di operazioni ed esistono diversi metodi di sagomare oggetti:

- Con i comandi: *!sagomal* , *!sagomaa*, *!sagomap* controlliamo l'orientamento dell'estrusione;
- FO: permette di definire i punti successivi come inizio foro;
- FR: permette di definire un foro rettangolare;
- FE: permette un foro ellittico;
- CE: un cerchio. Esempio: !sagoma TMP ce30,0,0,50 definisce un cerchio posizionato nel punto 0,0 e raggio 50.

Le sagome possono essere scritte in forma parametrica anche nell'archivio *sagome*, anche se è meglio all'interno della regola.

Possono essere utilizzate sagome non parametriche importate da *DXF* e/o utilizzando il CAD2D a disposizione.



Figura 3.51: Mobile sagomato.

3.10.7 Oggetti 3D esterni

3CadEvolution permette di utilizzare oggetti 3D creati con altri *CAD* e resi disponibili in vari formati, tra cui il formato *3DS*, il formato X e il formato DRG2 (dragotech) (anche WRL,e OBJ Wavefront).

Non sono disponibili interfacce dirette per i file in formato *DWG* (che comunque non sono molto adatti alla grafica 3D di ***3CadEvolution*** , mancando di mappatura dei materiali), ma possono essere convertiti utilizzando programmi di trasformazione di formato.

Le tre famiglie principali di oggetti riutilizzabili restano:

- 3DS: 3D Studio Max, Maya e la maggior parte di programmi di grafica 3D;
- X: DirectX e la maggior parte di programmi di grafica 3D;
- DRG2: derivazione X, ma aggiunte multimediali.

Gli oggetti *3DS* possono essere inseriti normalmente al posto di una normale tipologia. Creiamo un nuovo articolo _003 e lo associamo a una nuova regola `_003.ERG`, come abbiamo visto fare precedentemente.

Utilizzando la tipologia di Test (già vista precedentemente) abbiamo la possibilità di creare un parallelepipedo di dimensioni pari a quelle del BOX.

Codice 38. Tipologia di test.

```
001 // Test
```

Integriamo a questo punto il nome di un *3DS* (abbiamo già utilizzato delle maniglie precedentemente poste nella cartella `3DS/maniglie`, quindi scriviamo:

Esempio 30. Integrazione del nome di un 3DS.

Codice 39. Inserimento di un 3DS.

```
#LETS colbase=$(colore:3)|$(.percorso)\foto\legni\$(colore:4)
001,LATO=11,GL=$(.percorso)\3ds\maniglie\pomolora.3ds,c1=$(
    colbase),c2=$(colbase)
```

E' evidente che il file *3DS* viene sostituito al parallelepipedo di ingombro e che viene colorato con le mappature presenti nel file, utilizzando i normali colori 1 e 2. Qui occorre però fare attenzione al fatto che il file *3DS* originale può contenere n colori, e in questo caso occorre passare per uno script che faccia uso della funzione WR3D2 (appendice), oppure meglio passare per un oggetto X o DRG2.



Figura 3.52: Maniglia con utilizzo del 3DS esterno.

Gli oggetti possono essere inseriti utilizzando solo Script e differiscono dai 3DS in quanto:

- Non occorre dichiarare le dimensioni, perché queste sono all'interno dell'oggetto (l'oggetto può essere scalato utilizzando dei fattori di scala di default =1,1,1);
- occorre distinguere in oggetti distinti i file che contengono trasparenza da quelli che non la contengono e richiamare la funzione WRX2 (appendice) con i parametri corretti;
- possono avere materiali (anche tutti) memorizzati direttamente all'interno del file, da mescolare con materiali variabili (massimo 9).

Gli oggetti DRG2 sono di fatto delle vere e proprie funzioni che integrano al loro interno oggetti diversi e logiche di comportamento, oltre a movimenti e a effetti (musica, shader). La costruzione di un oggetto DRG2 è molto complessa ed è fatta esclusivamente dall'azienda, possono essere comunque utilizzati in ogni catalogo.

3.11 Regole con le aggiunte

Creiamo l'articolo _004 e la regola `_004.erg`, come fatto precedentemente.

A questo punto utilizziamo la regola `_004.erg` per costruire un BOX a vani. Il numero di vani, la loro dimensione e le loro varianti risultano a scelta dell'operatore.

In questo esempio la regola segue un percorso abbastanza diverso dal precedente in quanto nel gruppo di definizione aggiunge n istanze dei gruppi dichiarati come *aggiunta*, su specifica dell'operatore.

La regola viene valutata in questo modo:

- Si valuta il gruppo grafica che definisce la variabile _macro_finale che è il nome della macro da richiamare alla fine;
- per ogni aggiunta inserita, si valuta un gruppo il cui nome è lo stesso della dichiarazione dell'aggiunta;
- si valuta la macro definita da _macro_finale.
- Variante *"Altezza"* tipo libero, default 640;
- Condizioni di validità: abbiamo sostituito $(tipof:3)con $(tipof##:3)per gestire le n istanze del controllo. Questo modo di scrivere serve comunque solo nei gruppi di definizione.

A questo punto proviamo a far funzionare la regola, noteremo un pulsante + e un pulsante – che se premuti creano una nuova linguetta dopo aver fatto scegliere quale aggiunta vogliamo inserire (abbiamo solo un tipo di aggiunta).

Vediamo come ogni volta che si crea un'aggiunta, il programma crei una nuova istanza per ogni variante inserita. Per esempio *"altezza"* diventerà *"altezza_0000"*, e *"altezza_0001"*.

Adesso possiamo impostare la nostra regola. Vogliamo costruire un BOX che si dimensioni automaticamente per ogni aggiunta e per far questo impostiamo la regola come segue:

Esempio 31. Regola del box che si ridimensiona automaticamente per ogni aggiunta.

Codice 40. Box con le aggiunte.

```
1 //VanoAggiunta
2 //VanoFine
3 Assegna
4 #LET aTot=$(spessore)
5 #LETS _macrofinale=VanoFine
```

Le prime 2 righe commentate sono le macro che verranno richiamate, la prima per ogni aggiunta e la seconda alla fine della regola. La logica di funzionamento del programma per esteso sarebbe la seguente:

Codice 41. Logica di funzionamento delle aggiunte.

```
1 Assegna
2 #LET aTot=$(spessore)
3 VanoAggiunta,altezza=$(altezza_0000),tipof=$(tipof_0000),
    _passi=$(passi_0000) // 1a aggiunta
4 VanoAggiunta,altezza=$(altezza_0001),tipof=$(tipof_0001),
    _passi=$(passi_0001) // 2a aggiunta
5 VanoAggiunta,altezza=$(altezza_0002),tipof=$(tipof_0002),
    _passi=$(passi_0002) // 3a aggiunta
```

```
VanoAggiunta,altezza=$(altezza_...),tipof=$(tipof_...),_passi
    =$(passi_...) //Na-Esima aggiunta
VanoFine
```

Scrivere una regola con le aggiunte, è esattamente come scrivere il codice per esteso, salvo il fatto che si dovrebbe scrivere di più e non sarebbe comunque semplice gestire dichiarazioni variabili e un numero variabile di macro.

Tornando al primo pezzo di codice le prime due righe commentate non servono, ma inserendole in questo modo, quando si preme Ctrl+M sopra queste righe si entra immediatamente in editor della macro commentata e questo facilita le operazioni di manutenzione.

La variabile ATOT che definiamo pari a $(spessore), conterrà l'altezza finale del modulo ricavata sommando semplicemente le altezze di ogni vano, come possiamo vedere se modifichiamo la macro *VanoAggiunta* con il seguente codice: (premere CCtrl+M sopra la prima riga e inserire il codice riportato qui sotto).

Codice 42. VanoAggiunta.

```
// memorizza nuova altezza
#LET alt=$(altezza:0) // Altezza del vano
#IF $(passi:0)=
  #LETS passo=320;640;960;1280;1600;1920
#ELSE
  #LETS passo=$(passi:0)
#ENDIF

#IF $(tipof:4)=1
   #FOR $(passo),$(i)>$(alt)-100
     004,sa=$(i)+$(atot)
   #ENDFOR
#ELSEIF $(tipof:4)=2
   #FOR $(passo),$(i)>$(alt)-100
     004.001,sa=$(i)+$(aTot)
   #ENDFOR
#ENDIF
#IF $(tipof:1)=2
   _ante,sl=$(spessore),sa=$(atot),l=$l-$(spessore)*2,a=$(alt)
       -$(spessore),pman=$pm
#ELSEIF $(tipof:1)=3
   _anteV,sl=$(spessore),sa=$(atot),l=$l-$(spessore)*2,a=$(alt
       )-$(spessore),pman=$pm
#ELSEIF $(tipof:1)=4
   // Cassetti
   _cassetti,sl=$(spessore),sa=$(atot),l=$l-$(spessore)*2,a=$(
       alt)-$(spessore),pman=$pm
#ENDIF
#LET atot=$(aTot)+$(alt)
003,SA=$(atot)-$(spessore) // Ripiano
```

1. Definisce l'altezza del vano sulla variabile ALT, qui si potrebbe anche calcolare una dimensione minima o massima; è più comodo intervenire sulla variabile alt per tutto il codice successivo.

2. Qui si definisce una variabile passi per il posizionamento dei ripiani che se non dichiarata da operatore ($(passi:0)=0), viene impostata a elementi standard; il comando *#FOR*, visto precedentemente accetta un elenco valori separato da ;.
3. Si tratta di una rielaborazione del ciclo già impostato nella macro BOX, notare che è cambiato il controllo dalla variabile A alla variabile ALT, e lo spostamento dei ripiani in altezza è impostato sulla variabile ATOT.
4. Come nel punto 3 si tratta di una rielaborazione del controllo per posizionare ante, ante vetro o frontali, già visto nel punto precedente.
5. E' il cuore del funzionamento della macro: dopo averla inserita, ridefinisco la variabile ATOT, sommando la dimensione del vano appena inserito. A questo punto posiziono un ripiano di separazione del vano.

Possiamo provare questo codice e vediamo che la regola disegna ogni vano, ma manca ancora una cosa, la struttura: fianchi, schienali e base sono mancanti e devono essere gestiti dalla macro *VANOFINE*:

Codice 43. VANOFINE.

```
#LET a=$(atot)
002 // Fianco
002.001 // Fianco DX
003,sa=0 // Ripiano
005 // Schienale
010 // Interferenza
#IF $(vTop:1)=1
  020 // Top
#ENDIF
```

1. Definiamo l'altezza complessiva della regola.
2. Costruiamo la struttura intorno all'elemento selezionato.

Con questo abbiamo definito una regola contenente delle aggiunte. Possiamo fare alcune considerazioni:

- Non è possibile controllare direttamente il numero di aggiunte minimo o massimo, ma è possibile creare un articolo con n-Aggiunte e bloccare la possibilità di crearne e cancellarne di nuove;
- le aggiunte si inseriscono sempre alla fine, e quando si cancellano seguono il percorso inverso;
- è possibile selezionare tra più macro di aggiunta, nell'esempio abbiamo fatto una sola aggiunta con la macro vanoaggiunta, ma era comunque possibile definire una serie di macro (antaaggiunta, antavaggiunta e cassettoaggiunta, con funzionamenti diversi);
- ogni macro parla con la successiva tramite variabili comuni (es: ATOT).

In Figura 3.53 nella pagina 78 è rappresentato un esempio del risultato ottenuto, inserendo 3 vani.



Figura 3.53: Mobile con tre vani.

3.12 Regole con le divisioni

La gestione che abbiamo definito delle *divisioni*, permette di partizionare lo spazio Larghezza, Altezza attraverso un editor specifico e può richiamare per ogni Linea di Divisione e per ogni vano una macro posizionale.

Questo approccio permette di semplificare notevolmente regole che potrebbero diventare estremamente complesse, ma nello stesso tempo si tratta di un tool, appositamente creato per risolvere una serie di problemi specifici, quindi non è adatto a tutte le situazioni.

Le aggiunte hanno inserito un notevole grado di libertà alle regole, permettendo di avere un numero variabile di varianti, definite in funzione della complessità dell'oggetto. Tuttavia non è facile applicarle a una situazione di variabilità su più assi.

Creiamo l'articolo _005 e la regola `_005.erg`, come fatto precedentemente, utilizzeremo questa regola per creare una parete attrezzata variabile, quindi all'articolo 005 diamo dimensioni di default L=2500, a=2000, p=400.

Impostiamo il gruppo di definizione della regola, come nello schema visibile in Figura 3.54 nella pagina 79.

Osserviamo la riga divisore in cui sono impostate varie informazioni, separate da ,.

Aprendo l'editor (selezionando la riga e cliccando sul tasto Invio), apparirà la finestra, come in Figura 3.56 nella pagina 79:

Qui sarà possibile impostare tutti i parametri di personalizzazione del divisore:

- Macro aggiunta struttura: Codice della macro da richiamare per la creazione delle parti strutturali. Questa macro non ha varianti specifiche, quindi tutti gli elementi strutturali fanno riferimento alla stessa macro:

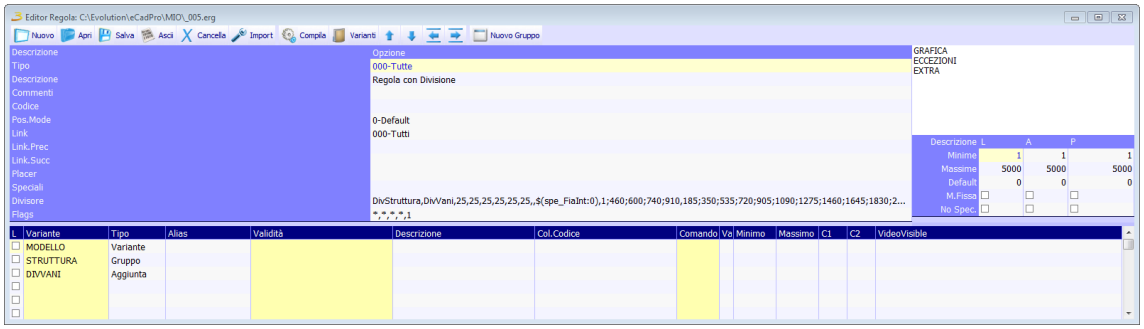


Figura 3.54: Regola con le divisioni, variante DIVVANI.

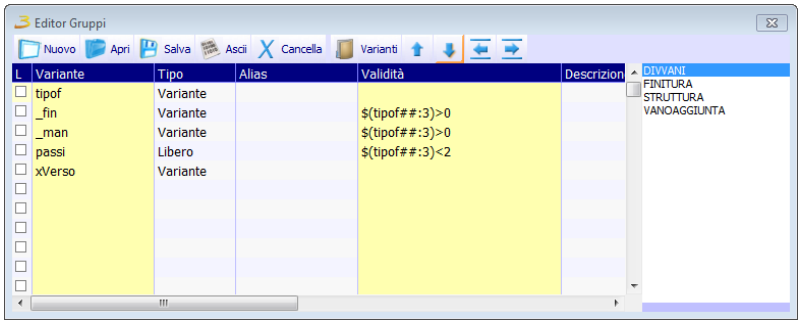


Figura 3.55: Editor gruppi, DIVVANI.

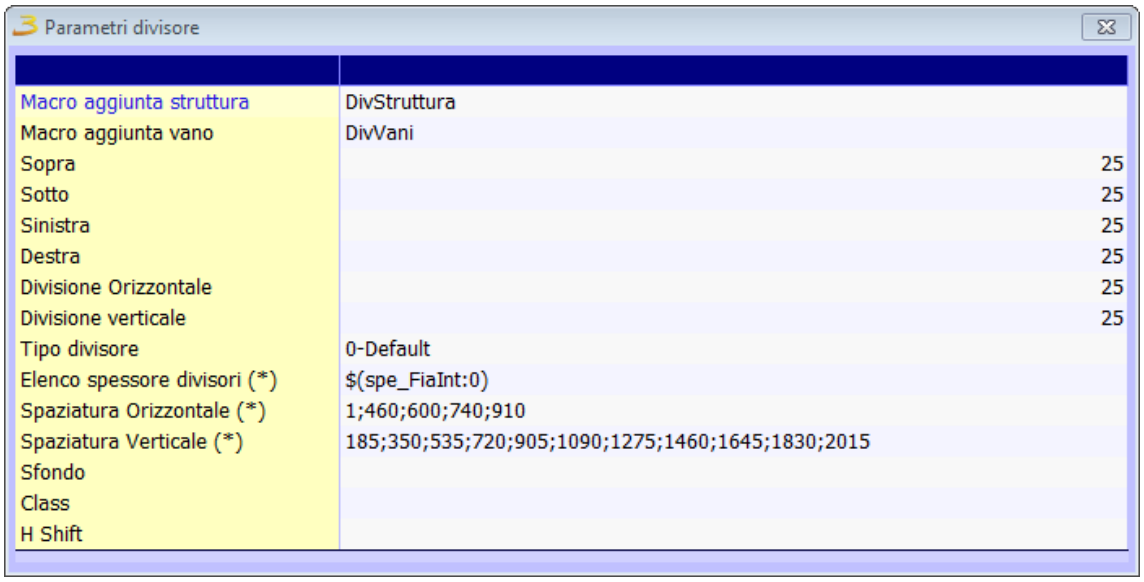


Figura 3.56: Parametri divisore.

abbiamo inserito noi del codice per identificare fianchi (verticali) rispetto a ripiani strutturali (orizzontali);

- Macro aggiunta vano: codice della macro da richiamare per ogni vano. Notiamo come nel rimando 2, questa macro sia riportata anche come aggiunta, in quanto funziona esattamente come le aggiunte del paragrafo precedente;
- Sopra, Sotto, Sinistra, Destra: con questi parametri è possibile indicare le dimensioni che riguardano eventuali cornici, zoccoli, fianco destro e sinistro. Nella Figura 3.56 nella pagina 79 si può notare che viene assegnata una variabile per le dimensioni del fianco sinistro e per quello destro;
- Divisione orizzontale, Divisione verticale: dimensioni delle divisioni;
- Tipo divisore: [3] ;
- Elenco spessore divisori: elenco dei vari spessori dei divisori separati da [;] ;
- Spaziatura orizzontale, Spaziatura verticale: qui possiamo elencare i vari passi a cui farà riferimento lo spostamento manuale delle divisioni sempre divisi da ;. Per l'inserimento di una misura diversa da quelle elencate usare la barra spaziatrice;
- Sfondo: applica uno sfondo alla sezione riguardante il divisore;
- Class: [4];
- H Shift: si tratta di un delta in altezza.

La macro *DivVani* è stata dichiarata come aggiunta, e nella parte operativa è stata inserita una nuova variante per il verso di apertura dell'anta, per ogni vano, visto che per questo tipo di regola non è sufficiente utilizzare la variabile di sistema $pm, che è legata al vano.

Procediamo a questo punto con la creazione del gruppo di valutazione, che è strutturato in 3 parti.

La prima parte, è quella generica della regola e definisce la struttura esterna della stessa.

Questo gruppo è praticamente identico alle situazioni precedenti in quanto va semplicemente ad assegnare le variabili di sistema, inserisce fianco sinistro, destro, fondo e cielo.

Codice 44. DivStruttura e DivVani.

```
// DivStruttura
// DivVani
assegna
#LET aperto=0
```

[3] In fase di sviluppo.
[4] In fase di sviluppo.

```
#LET spessore=25
002 // Fianco
002.001 // Fianco DX
003 // Ripiano
003,SA=$a-$(spessore)// Ripiano
```

Vediamo la macro DivStruttura che è molto semplice:

Codice 45. Macro DivStruttura.

```
#IF $l<$a
   002 // Fianco
#ELSE
   003,SL=0,L=$l // Ripiano
#ENDIF
```

Qui abbiamo controllato la dimensione del vano creato per DivStruttura, infatti all'interno di questa macro vediamo $a e $l non come le dimensioni complessive del box, ma solo come le dimensioni della parte da riempire e questo facilita di molto le cose.

La macro DivVani invece è simile a quella utilizzata per le aggiunte:

Codice 46. Macro DivVani.

```
#IF $(passi:0)=
  #LETS passo=320;640;960;1280;1600;1920
#ELSE
  #LETS passo=$(passi:0)
#ENDIF

#IF $(tipof:4)=1
   #FOR $(passo),$(i)>$a-100
      004,sa=$(i),sl=0,l=$l
   #ENDFOR
#ELSEIF $(tipof:4)=2
   #FOR $(passo),$(i)>$a-100
      004.001,sa=$(i),sl=0,l=$l
   #ENDFOR
#ENDIF

#IF $(tipof:1)=2
    005,sl=0,sa=0,l=$l,a=$a // Schienale
   _ante,sl=0,sa=0,l=$l,a=$a,pman=$(xVerso:1)
#ELSEIF $(tipof:1)=3
    005,sl=0,sa=0,l=$l,a=$a // Schienale
   _anteV,sl=0,sa=0,l=$l,a=$a,pman=$(xVerso:1)
#ELSEIF $(tipof:1)=4
   005,sl=0,sa=0,l=$l,a=$a // Schienale
   _cassetti,sl=0,sa=0,l=$l,a=$a // Cassetti
#ENDIF
```

1. Il controllo sulle altezze e sulle larghezze viene fatto sulle variabili di sistema $a e $l.

2. Abbiamo inserito uno schienale per vano e solo per i vani con anta e cassetto.
3. Le macro _ante,_anteV e _cassetti sono posizionate in 0,0 e hanno dimensione $l e $a, come specificato anche al punto 1.
4. Il verso di apertura delle ante è stato gestito per vano.

Utilizziamo l'oggetto divider per creare una rappresentazione schematica dell'oggetto che andremo a produrre per ottenere un risultato simile alla Figura 3.57.

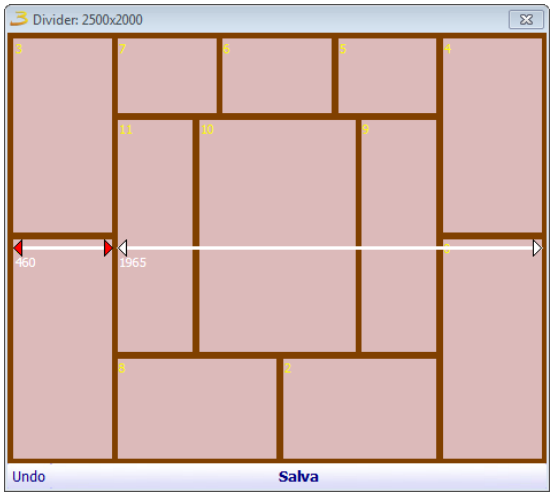


Figura 3.57: Oggetto devider.



Figura 3.58: Mobile con utilizzo dei divider.

Una regola con divisioni ha potenzialità molto elevate e permette spesso di gestire con poco codice anche situazioni molto complesse.

3.13 Regole con il placer

Con questa modalità, sarà più semplice gestire i frontali partendo da una regola con le divisioni.

Per semplicità indicheremo la regola principale della divisione come padre e le regole dei frontali come figli.

Innanzitutto 24 è il numero del lato che identifica la tipologia placer. Oltre a questo, è indispensabile specificare il tipo di placer nel parametro X1.

Nell'editor della regola, aggiungere nell'apposita sezione *Placer* le tipologie di placer elencate all'interno delle tipologie. È possibile indicare diversi tipi di placer, separati dal carattere ▭, per esempio:

123.

Ci sono diverse variabili da utilizzare all'interno della regola del placer. Le principali sono:

- _PLACER: vale 1 quando il placer viene abilitato;
- _XDA, _XDL, _XDP: dimensioni del divisore totale;
- _PPA, _PPP, _PPL: dimensioni rispetto all'elemento completo.

E' possibile passare dei valori dalla regola del figlio alla regola del padre attraverso delle variabili di sistema. In particolare per passare un valore dai figli al padre si usa LOC_BXn° , dove n° può andare da 1 a 9. Con !PLACERVAR nome_variabile n° che andrò a richiamare nel padre, posso richiamare la variabile specificata precedentemente nel figlio.

Nel caso contrario, ovvero richiamare una variante del padre ai figli, usare il comando LOCP_BXn° con la stessa sintassi elencata sopra.

Nei casi in cui è necessario forzare la rivalutazione dei figli o del padre, indicare.

Esempio 32. Forzare la rivalutazione dei figli per i placer.

Codice 47. Parametro _PLACERRIVFIGLI.

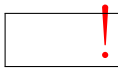
```
1 #LET _PLACERRIVFIGLI=1
```

oppure,

Esempio 33. Forzare la rivalutazione del padre.

Codice 48. Parametro _PLACERRIVPADRE.

```
1 #LET _PLACERRIVPADRE=1
```



Evitare di usare questi due comandi insieme.

3.14 Placer multicatologo

Questa funzionalità pur rifacendosi al placer tradizionale introduce diversi cambiamenti e permette di inserire oggetti che non devono stare necessariamente nello stesso catalogo.

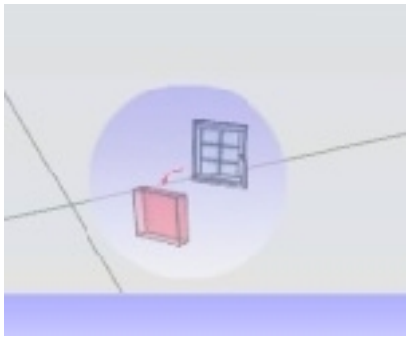


Figura 3.59: Il simbolo in grafica del placer multicatalogo.

Il programma non attiva i placer multicatalogo in automatico, ma fa apparire sul box selezionato una icona in basso . Questa icona, 3.59 dice che esistono nel box selezionato oggetti inseribili. premendola appare un menu di scelta per ogni oggetto inseribile con il contenuto della tipologia. Se non si vuole inserire nulla basta premere il tasto Esc.

Il funzionamento:

- La regola genitore deve contenere uno o più riferimenti a una tipologia con *lato 43* (placer multicatalogo);
- questa tipologia rappresenta un ingombro posizionato in un punto (x,y,z) del box e di dimensioni prefissate. All'interno di questo sarà contenuto il box sostitutivo;
- nel parametro 1 della tipologia si deve integrare il riferimento a uno o più oggetti inseribili: questo rappresenta un cambiamento rispetto al placer tradizionale in cui gli ingombri del placer contenevano solo un riferimento al tipo di oggetto che si voleva inserire;
- i riferimenti sono separati tra di loro con il simbolo */.* e sono impostati con un separatore *;* in 4 campi:
  1 - codice dell'articolo inseribile
  2 - modello dell'articolo inseribile (se vuoto usa il modello corrente, se su un altro catalogo e vuoto usa il modello 999
  3 - catalogo (se vuoto usa il catalogo corrente)
  4 - descrizione ai fini del *PopUp* menu di scelta per capire cosa si vuole inserire.

Esempio 34. Definizione di un placer multicatalogo.

Codice 49. Placer multicatalogo.

```
237, lato=43, x1=_007;999;DauWall;Ante /_005;999;DauWall;
   Cilindro // Placer MultiCatalogo
```

3.15 Macro di più articoli

Con questa funzionalità, accessibile attraverso il tasto destro del mouse in grafica, la Macro Regola permette di legare tra loro diversi oggetti per formare un unico codice composto da diversi oggetti (ognuno col codice originario).

Prendiamo per esempio una situazione in cui si vuole concatenare 3 elementi.

Partiamo col richiamarli in grafica, dopodiché nel menu a tendina del tasto destro selezioniamo Macro Regola. All'apertura della finestra come in Figura 3.60 nella pagina 85, procedere con il nominare la macro e dando un'eventuale descrizione, selezionare i modelli su cui rendere accessibile questo nuovo elemento e la tipologia nelle associazioni in cui includerlo. Salvare per rendere accessibile le variazioni.

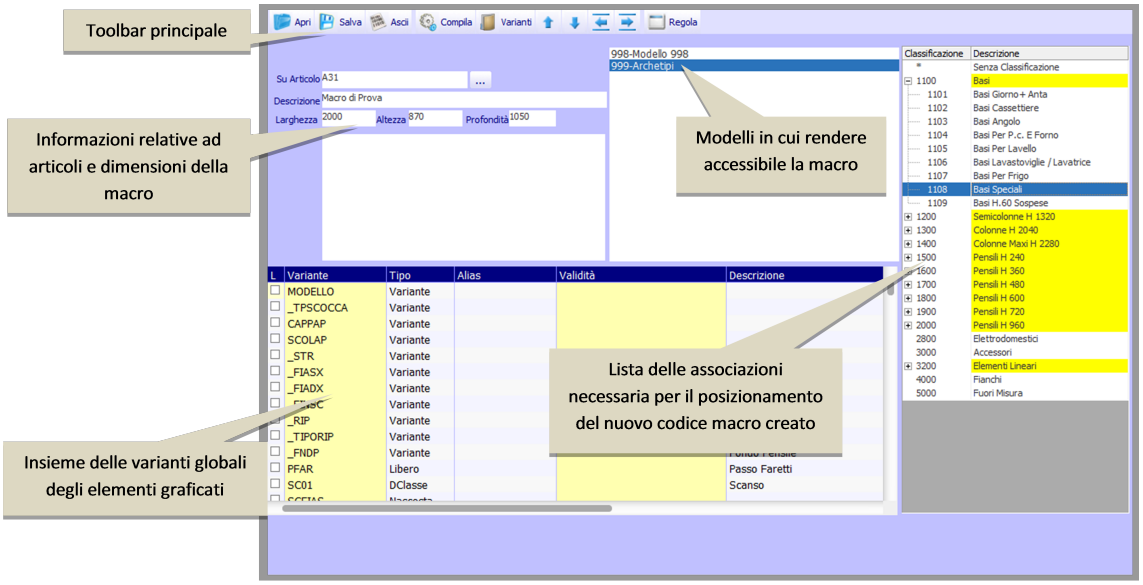


Figura 3.60: Macro di più articoli.

La regola grafica sarà gestita in questo modo:

Codice 50. Regola con le macro di più articoli.

```
REG 999_BA1CS2/$(_STR:@);$(_FIASX:@);$(_FIADX:@);$(_MAN:@);\
$(_GUI:@);$(_FIN:@);$(_FIN2:@);$(_FIN3:@), \
$_BA1CS2,0,80,0,0,LATOSX=0,LATODX=0,LATOASX=0,LATOADX=0,
    LATOBSX=0,\ LATOBDX=0,PM=3, \
    L=600,A=720,P=560, \
    _TPSCOCCA=BC,_FRO=16
#REG 998BB1500/$(_STR:@);$(_FIASX:@);$(_FIADX:@);$(_MAN:@);\
    $(_FIN:@), \
    $_B1A,600,80,0,0,LATOSX=0,LATODX=0,LATOASX=0,LATOADX=0,\
        LATOBSX=0,LATOBDX=0,PM=3, \
    L=150,A=720,P=560, \
    _TPSCOCCA=BA,_RIP=0,_FRO=01
#REG 999_SC5CA1A/$(_STR:@);$(_FIASX:@);$(_FIADX:@);\ $(_MAN:@
    );$(_GUI:@);$(_FIN:@);$(_FIN2:@);$(_FIN3:@);\ $(_FIN4:@);$
    (_FIN5:@);$(_FIN9:@), \
    $_SC5CA1A,750,80,0,0,LATOSX=0,LATODX=0,LATOASX=0,LATOADX
        =0,\ LATOBSX=0,LATOBDX=0,PM=3, \
    L=600,A=1320,P=560, \
    _TPSCOCCA=CN,_RIP=0,FRONTALI=01, \
    _FRO=32,_FRO1=01,_FIN9=1.06
```

```
15 #LET L=1350,A=1400,P=560,NL=0,NA=0,NP=0,UL=0,UA=0,UP=0
```

Qui possiamo notare che, tramite il comando *#REG*, annotiamo tutte le informazioni principali degli oggetti all'interno della regola (codice articolo, varianti grafiche, dimensioni) e come ultimo comando, il *#LET* che gestisce le dimensioni e i tagli. Il tutto è gestito e creato automaticamente.

Inoltre, nel file *ecadpro.ini* all'interno del catalogo, esistono due flag: - MACROVARIANTIBASE= e MACROVARIANTIOBB=, che servono ad includere o ad obbligare delle varianti in fase di costituzione della macro. Basta semplicemente aggiungere dopo = le varianti interessate divise da ,.

Una volta caricato l'articolo nuovo, non sarà possibile effettuare delle modifiche per singolo elemento, perché saranno considerati un unico oggetto. L'unico modo per farlo, è fare tasto destro in grafica e selezionare la voce Dividi dal menu a tendina. In questo modo la macro verrà spezzata e gli elementi torneranno nella configurazione originale.

3.16 Distinta base a formule e listino a formule

Dopo aver creato nei paragrafi precedenti alcune regole procediamo ora creare dai disegni una distinta base dei pezzi necessari per produrre gli elementi disegnati.

Per fare questo è necessario creare una anagrafica dei semilavorati; lo strumento più semplice da utilizzare per questo caso è il generatore di codici, ma possiamo anche procedere con l'inserimento nell'anagrafica articoli di tutti i componenti di cui abbiamo bisogno.

Abbiamo bisogno sostanzialmente di una anagrafica di tutti i componenti di base che vorremo avere come risultato dal disegno. Tale anagrafica può avere diversi sistemi di codifica in base a come l'azienda di produzione opera.

Per esempio, per un anta, è possibile codificare tutti i codici dati dalla combinazione delle dimensioni di acquisto per i colori utilizzati, oppure codificare un'anta diversa solo per dimensione o solo per colore o ancora un codice unico per tutte le ante.

Per ***3CadEvolution*** cambia poco avere a che fare con migliaia di codici o pochi codici, in quanto è possibile comunque identificare tutte le caratteristiche di un componente attraverso informazioni grafiche (in particolare legate al fatto che a ogni oggetto disegnato sia legata una matricola univoca, che permette di ricavare le varie informazioni utili alla produzione, quali disegno tecnico, forature, lavorazioni).

Di fatto però, la produzione necessita spesso di avere un maggior dettaglio possibile, quindi richiede codici diversi per gestire costi e lavorazioni. Supponiamo per esempio di acquistare 4 misure di Ante 150,300,450,600 se nel disegno necessita di un'anta di dimensione 500, dovranno essere fatti alcuni ragionamenti come ad esempio quello di prendere l'anta di dimensione superiore e attivare una lavorazione aggiuntiva di taglio.

Gestiamo questi criteri tramite il CodiceBarra.

Prima però vogliamo definire una serie di reparti di produzione in modo che questi codici siano suddividibili nelle varie stampe (per poter ad esempio separare ferramenta, da ante e da elementi strutturali).

Andiamo in Gestione delle tabelle e inseriamo la tabella RP (Reparti) come in Figura 3.61:



Figura 3.61: Gestione delle tabelle, tabella RP.



Figura 3.62: Inserimento in anagrafica articoli dei codici.

A questo punto inseriamo in anagrafica degli articoli i seguenti codici articolo e la divisione in reparti, Figura 3.62.

Possiamo codificare ogni singolo articolo della lista utilizzando l'anagrafica degli articoli messa a disposizione.

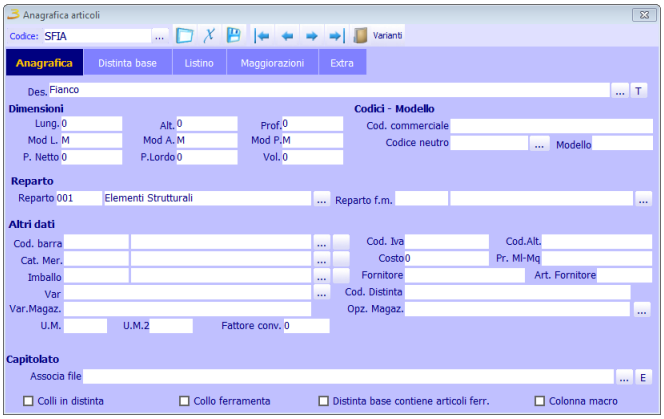


Figura 3.63: Modifica articolo, SFIA.

Qui si può vedere come il disegno in realtà contenga già una lista di componenti che sono quelli effettivamente disegnati.

Se noi prendiamo il disegno di un Box, presente in Figura 3.64, (carico l'articolo 001), guardiamo il dettaglio della regola e troviamo uno schema come in Figura 3.65.



Figura 3.64: Disegno del Box, articolo 001.

Funzioni

Articolo | Associazioni | **Dettaglio**

Nuovo | Cancella | Duplica | Redraw | Sagoma | - | Blocca

Tipologia	Lato	PL	PA	PP	DL	DA	DP	AXY	AXZ	AYZ	C1
002-Fianco	0	0	0	0	25	900	400	0	0	0	10|MIO\FOTO\LEGNI\03.jpg
002-Fianco	0	275	0	0	25	900	400	0	0	0	10|MIO\FOTO\LEGNI\03.jpg
003-Ripiano	0	25	0	0	250	25	400	0	0	0	10|MIO\FOTO\LEGNI\03.jpg
003-Ripiano	0	25	875	0	250	25	400	0	0	0	10|MIO\FOTO\LEGNI\03.jpg
005-Schienale	0	25	25	10	250	850	10	0	0	0	10|MIO\FOTO\LEGNI\03.jpg
004-Ripiano Interno	0	25	450	20	250	25	360	0	0	0	10|MIO\FOTO\LEGNI\03.jpg
010-Interferenza	23	0	0	0	300	900	400	0	0	0	0
020-Top	0	0	900	0	300	10	410	0	0	0	10|MIO\FOTO\LEGNI\03.jpg

Figura 3.65: Dettaglio della regola rappresentante il Box.

Tali componenti sono semplicemente le tipologie, ma spesso è difficile ricavare da questi in modo diretto tutte le informazioni e vi sono oggetti quali l'interferenza che è utilizzata solo internamente, ma non ha alcuna relazione con gli elementi da produrre.

Possiamo comunque pensare di legare uno o più codici a ogni tipologia inserita, utilizzandola in modo parametrico fino a ottenere l'elenco esatto e completo di quello che ci serve per la produzione.

Nel caso del Fianco per Esempio andremo a scaricare il codice del fianco + 2 Piedini.

Entriamo nella Tipologia '002' e inseriamo la distinta, come una regola, per l'impostazione dei campi di ogni riga premiamo il tasto Ctrl+H.

Prima di procedere con ogni spiegazione ulteriore, passiamo a controllare il disegno di sopra e a verificare la distinta base creata.

Possiamo vedere che sono stati scaricati 2 Fianchi e quattro piedini e che per i fianchi è stata impostata la dimensione.

Spieghiamo la riga:

Codice 51. Regola per anagrafica articoli.

```
1 SFIA,1,,$(da),$(dp),$(dl)
```

Qui sono state utilizzate delle variabili di sistema, utili per la distinta base: come abbiamo premesso la Distinta a formule non è calcolata nello stesso momento in cui viene valutata la regola, ma interpreta il contenuto della tipologia, impostando ogni informazione contenuta su variabili e utilizzando queste variabili per decidere cosa scaricare come informazione.

3.17 Distinta base estesa

Il configuratore di ***3CadEvolution*** gestisce con la grafica un disegno cui associa una distinta base di componenti, tramite le funzioni di distinta base grafica. Ad ogni elemento di disegno è associata una tipologia che permette di descrivere completamente l'elemento attraverso un numero variabile di parametri.

A partire da ogni tipologia, interpretando questi parametri, il programma è in grado di ricavare un elenco di componenti utilizzando funzioni estese come il calcolo dei codici da codice barra ecc.

La limitazione di questo approccio (salvo artifici) è che è fatto per gestire solo il primo livello di distinta base, dopo di che il calcolo viene demandato al sistema gestionale.

La distinta base estesa, attraverso una serie di funzioni scritte all'interno del database, permette di gestire con il configuratore tutti i livelli di distinta base, completi di cicli e fasi. Il concetto applicato è il seguente:

- Ad ogni articolo viene associata una distinta base, che altro non è che una specie di regola i cui parametri sono presi dalle varianti dell'articolo stesso;
- a completamento delle varianti dell'articolo vi sono delle forzature, ossia varianti i cui valori sono strettamente determinati dall'articolo stesso ma che sono interrogabili all'interno della distinta base;
- ogni distinta ha diverse versioni di validità e l'utilizzo di una distinta piuttosto che un'altra può essere determinato tramite il valore di varianti, il fuori misura, la quantità da produrre, il cliente, in base al risultato di una funzione di database;
- ogni distinta base è fatta per gestire un elenco di componenti attribuite in fasi diverse di lavorazione: poiché il magazzino è in grado di distinguere le giacenze per fase non è necessario creare un numero elevato di componenti intermedi di trasformazione: ad esempio il materiale laccato può essere lo stesso codice del materiale da laccare, visto che la laccatura stessa è una fase di lavoro.

La gestione della distinta base, prevede una tabella delle fasi di lavorazione (procedura gestione delle fasi) in cui per ogni singola fase sono indicati i costi (divisi in 3 parametri di calcolo) .

Nell'esempio di distinta base estesa, contenuto nel catalogo "DAUWALL", tutto il materiale pannelli, utilizza la stessa distinta base estesa, divisa in due versioni, una per lo standard e una per il fuori misura. Vediamo come gli articoli, le quantità, le varianti e le note sono associate a una serie di variabili che ne determinano l'utilizzo.

Le variabili sono gestite nel formato [nomevariabile]; oppure nel formato tipico della regola: $(nomevariabile:colonna).

Tutte le varianti diventano variabili per la distinta base, inoltre sono automatiche le variabili L, A, P(dimensioni larghezza, altezza a profondità reali) e LL, AA, PP(dimensioni larghezza, altezza e profondità anagrafiche).

3.17.1 Installazione distinta base estesa

L'installazione della distinta base estesa comporta l'installazione di una *DLL* sul ***SQL Server*** . Questa *DLL* è compatibile solo con la versione 2000 di SQL server e il *Setup* va lanciato dalla macchina *Server*.

Occorre copiare i file *DBESTESA.SQL*, *PRODUZIONE.DDF* e *PRODUZIONE.SQL* nella cartella `_ECADPRO` e quindi dal menu setup attivare il tasto PARIFICA PROD. ESTESA in questo modo vengono copiate nel database tutte le funzioni per l'installazione della DBE. Il programma registra su *ECADPRO.INI* il flag produzioneEstesa=1 (per tornare alla gestione normale occorre semplicemente riportarlo a 0).

Per quanto riguarda l'installazione della produzione estesa con ***SQL Server*** 2005 e 2008 è stata inserita una nuova *DLL DauBaseSQLN.dll* che va registrata sul server.

I file necessari per questa versione della produzione estesa vanno copiati nella cartella `_ECADPRO` e sono i seguenti:

1. *caumag.TXT*;
2. *produzione.DDF*;
3. *produzione.SQL*;
4. *DauBaseSQLN.DLL*;
5. *dbEstesaN.SQL*;
6. *DatiProduzione.SQL*;
7. *dbEstesa.SQL*.

Per lanciare l'aggiornamento è sufficiente aprire l'***EcadUtil***, selezionare il database da aggiornare e premere il bottone Parifica produzione estesa, in questo modo la *DLL* viene copiata sul server e registrata, e il database viene aggiornato.

3.17.2 Gestione derive varianti

E' possibile gestire le derive di varianti tra padre e figlio indicando nella colonna varianti della distinta base la variante del padre con il suo valore, e la variante che deve essere assegnata al figlio.

La sintassi da usare è la seguente:

VARPADRE=OPZPADRE:VARFIGLIO=OPZFIGLIO oppure VARPADRE=*:VARFIGLIO=*

Ciò significa che al valore della variante del padre OPZPADRE, nell'articolo figlio verrà indicata la variante VARFIGLIO con valore OPZFIGLIO. L'asterisco indica tutti i valori. Perché questa gestione funzioni sull'anagrafica articoli del figlio dev'essere indicata tra le varianti VARFIGLIO.

4 Progettazione su regole

Una regola grafica è composta da una sequenza pilotata di riferimenti alle tipologie elementari, le quali devono essere state precedentemente definite nel database attraverso gli appositi strumenti forniti da ***3CadEvolution*** .

Una tipologia elementare, una volta richiamata, è in grado di produrre contemporaneamente l'immagine grafica e il codice dell'oggetto.

4.1 Definizione della regola

3CadEvolution mette a disposizione dell'operatore un particolare ambiente dedicato alla generazione delle regole. Vi si accede dal menu di disegno premendo il tasto Ctrl+R , dopo aver inserito un articolo.

Ogni regola è un file salvato in formato *ASCII* con estensione *ERG*, e compilato in un file con estensione *ERX*.



Figura 4.1: Modifica regola, pannello principale.

Dalla barra degli strumenti dell'editor delle regole possiamo creare una nuova regola, aprire, salvare, vedere la regola in formato *ASCII*, cancellare e compilare le regole.

La compilazione delle regole è una procedura, che deve essere fatta quando si fanno interventi con editor diversi alle regole, alle varianti, al database, questo perché tutte le informazioni vengono trasformate in un formato più efficiente per il linguaggio delle regole. La compilazione è automatica utilizzando gli strumenti standard a disposizione.

La regola è divisa in due parti: *Definizioni* e *Gruppi di Valutazione*:

- *Definizioni*: si definiscono le caratteristiche generali della regola come il suo nome, il gruppo di appartenenza, le tabelle di dimensioni etc.;
- *Gruppi di Valutazione*: si utilizza il linguaggio delle regole per definire quali sono i componenti base dell'elemento (tipologie), come sono disposti uno rispetto all'altro, il loro colore etc. Il gruppo di valutazione è un vero e proprio ambiente di programmazione.

Dal punto di vista tecnico entrambe le parti, modificabili tramite la finestra Edit, vengono poi salvate all'interno di un unico file testo o ascii con estensione ERG, in cui le varie sezioni sono contenute in gruppi diversi. E' possibile accedere al file ASCII della regola tramite l'apposito pulsante nella barra degli strumenti. Il relativo file compilato ha estensione *ERX* e si crea automaticamente quando si esce dall'Edit. Le regole si possono generare con qualsiasi editor di testo, in ogni caso, prima di essere utilizzabili nella costruzione dell'ordine con***3CadEvolution*** devono essere compilate usando il pulsante COMPILA presente nella barra degli strumenti.

Definizioni: La scheda in Figura 4.2 nella pagina 93 consente l'inserimento dei dati di base di una regola; vediamo in dettaglio tutte le voci contenute.

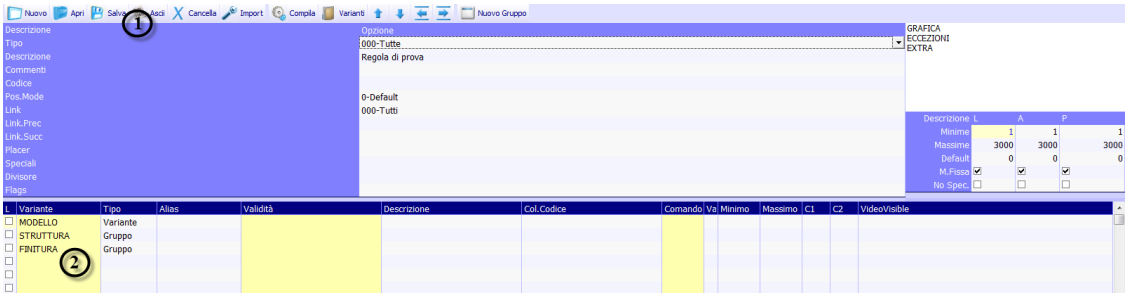


Figura 4.2: Editor regola.

(1) *Intestazione della regola.*
(2) *Elenco delle variabili usate dalla regola.*

- Tipo: si seleziona il gruppo di appartenenza della regola, corrispondente ad un codice della tabella TP-Tipo regole (come visto nel capitolo 1), precedentemente caricato. Per scegliere il tipo di regola si fa clic sulla freccia a destra che apre la casella di riepilogo;
- Descrizione: è la descrizione della regola; viene visualizzata nell'elenco delle regole nella fase di costruzione dell'ordine;
- Commenti: sono delle righe aggiuntive di descrizione o commento alla regola;
- Codice: permette di inserire un codice calcolato direttamente dalla regola. Se il campo rimane vuoto non modifica la gestione esistente, ma se inserito, determina il codice di preventivo;
- PosMode: indica il posizionamento della regola in grafica. Può assumere i valori riportati nella apposita tabella.
- Link: si tratta della linea di collocamento della regola. E' utilizzato solo in cucine dove c'è l'esigenza di posizionare ad una certa quota alcuni mobili come i pensili. E' legato ad una tabella di valori impostata nelle altezze dei flag speciali (v. paragrafo Configurazioni particolari del catalogo);
- Link Precedente/Link Successivo: si tratta del tipo di collegamento del box corrente, rispetto al box precedente e rispetto al box successivo. Il tipo di collegamento chiamato Link corrisponde ad un numero della tabella dei Link, e ci serve per posizionare il box della regola corrente, in una certa posizione relativamente al box precedente o successivo. Questo numero è il primo di 5 parametri che devono essere inseriti obbligatoriamente in queste due righe, se diversi da zero e separati da [,]. Gli spostamenti in Larghezza, Altezza e Profondità servono per comunicare a ***3CadEvolution*** di posizionare il nuovo box secondo la collocazione stabilita dal Tipo Link, ma con degli spostamenti aggiuntivi in larghezza e/o altezza e/o profondità che inseriamo usando questi parametri; spostamento angolare serve per posizionare il nuovo box con una certa rotazione che verrà sommata a quella eventualmente prevista

dal Tipo Link. Lo spostamento angolare è in senso antiorario. Il numero di link è sempre formato da 3 cifre numeriche. E' conveniente utilizzare la funzione *!ADDLINK* e non gestire questi campi che hanno una limitata funzionalità;

- Link sx Prec. e Succ.: vale quanto detto precedentemente per i Link. La doppia definizione è necessaria per gli angolari, soprattutto in cucine, nel qual caso l'elemento è collegato ad un box sia a dx che a sx, ma come già detto, è opportuno usare gli ADDLINK;
- Speciali: indica una serie di informazioni per istruire la regola a comportarsi in un certo modo: sono da utilizzarsi su indicazioni specifiche e determinano comportamenti particolari del programma;
- Divisore: se si impostano questi campi la regola funziona come un divisore (vedi capitoli precedenti);
- Flags: in questo campo impostiamo la visibilità per le variabili di completamento della regola che di default sono rappresentate in questo modo nella zona incorniciata nella Figura 4.3 nella pagina 95. Sono in realtà 7 check box, che definiscono altrettante variabili:
  - Zoccolo SX: $(latosx)
  - Zoccolo DX: $(latodx)
  - Alzatina SX: $(latoasx)
  - Alzatina DX: $(latoadx)
  - Cornice SX: $(latobSX)
  - Cornice DX: $(latobdx)
  - Aperto: $(aperto)

  Queste variabili hanno valore 0 se il box non è selezionato altrimenti 1; le prime 6 variabili vanno a coppie e l'ultima è a se stante.
  Oltre a questo vi è la variabile per il verso di apertura che corrisponde alla variabile $pm (1 se sx, 3 se dx).
  Detto questo nel campo flag è possibile nascondere o cambiare il nome che l'utente vede regola per regola: si scrive, separato da , la descrizione di ogni campo e un * se questo deve essere rimosso.
- Immediatamente sotto alla sezione dei gruppi di valutazione troviamo la tabella dei campi di variabilità delle dimensioni visibile in Figura 4.4 nella pagina 95: si tratta, in termini pratici, delle dimensioni massime e minime impostabili dall'operatore per la regola, e delle dimensioni di default proposte al momento dell'utilizzo della regola stessa. Quando l'elemento viene selezionato all'interno del relativo elenco della finestra di Costruzione, vengono proposte le dimensioni di default qui predefinite; l'operatore può comunque variare le misure dell'articolo mediante lo stiramento o la selezione nei relativi campi L, A, P. I valori massimi e minimi che l'utente potrà scegliere saranno quelli stabiliti in questa tabella. Si ricordi che l'ordine dei dati è sempre L, A, P ossia larghezza, altezza e profondità. Settando il flag misura fissa per una o più dimensioni, in fase di costruzione l'operatore non avrà a disposizione la scelta delle dimensioni caratteristiche, in quanto vengono bloccate le relative tabelle, potrà solo digitare dei valori fissi. Il flag no speciali permette di attribuire alla regola solo le misure contenute nelle tabelle di riferimento, ma non

consente di immettere dei valori differenti, in pratica vieta il cosiddetto fuori misura.
Questa gestione può comunque essere ritenuta obsoleta in quanto è possibile specificare meglio le stesse informazioni anche su ogni singolo articolo attraverso la tabella delle causali fuori misura, Figura 4.5 nella pagina 97.



Figura 4.3: Flag, variabili di completamento.

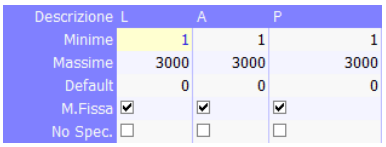


Figura 4.4: Campi di variabilità delle dimensioni.

Tipi di posizionamento della regola in grafica

PosMode	Descrizione	Definizione
0	Posizionamento Standard	Posizione attuale.
1	Posizionamento per Complementi	Il posizionamento della regola avviene sulla base degli elementi che incontra (vedi esempio vaso su base). Con Shift+freccia la regola si sposta in direzione della freccia legandosi alla regola su cui è posizionato. Attenzione: le collisioni per questo tipo di oggetto non possono essere attivate.
2	Tracciamento	Esiste ancora il modo di definire la regola sulla base dell'iniziale del nome (ZTP...) della regola per compatibilità, ma questo è il modo standard di definire i tracciamenti.
11	Cilindro no Angolo	Per poter utilizzare questo particolare tipo di tracciamenti è necessario inserire nello script della regola padre una tipologia LATO 15 che avrà come altezza e profondità il diametro del cilindro che vogliamo disegnare.
12	Cilindro Poligono	Questa tipologia fungerà da Placer per tutti gli oggetti che andremo ad inserire.

Tabella 4.1 : continua nella prossima pagina

Tabella 4.1 : continua dalla pagina precedente

PosMode	Descrizione	Definizione
13	Cilindro 2 Punti	Questi tracciamenti si comportano come dei Placer a tutti gli effetti dato che rimangono legati al padre e si possono spostare o duplicare liberamente rimanendo comunque collegati al padre.
14	Cilindro Muro Pavimento	è possibile inserire manualmente le misure del posizionamento premendo la barra spaziatrice.
15	Cilindro Goniometro	Si possono aggiungere i passi di spostamento inserendoli nel parametro X1 della tipologia padre.
16	Complementi Orizzontali	Come PosMode 1 (vedi esempio tappeto nel catalogo "COMPLIMENTI")
17	Complementi Verticali	Il posizionamento della regola avviene sulla base degli elementi verticali che incontra (vedi esempio specchio su muro nel catalogo "COMPLEMENTI").
18	Area di Vestizione	Si tratta di una funzione molto simile al placer. A differenza di quest'ultimo l'area di vestizione permette di posizionare e muovere liberamente l'oggetto nello spazio delimitato dalla tipologia.
19	Area di Vestizione X	Come PosMod 18, permette di posizionare l'oggetto solo in larghezza.
20	Area di Vestizione Y	Come PosMod 18, permette di posizionare l'oggetto solo in altezza.
21	Area di Vestizione (-)	Come PosMod 18 ma permette il posizionamento di un oggetto in negativo rispetto allo spazio delimitato dalla tipologia (esempio: cassetto all'interno di una base).
22	Area di Vestizione (-X)	Come PosMod 21 ma solo in larghezza.
23	Area di Vestizione (-Y)	Come PosMod 21 ma solo in altezza.
24	Floor Follower	In fase di sviluppo.

Tabella 4.1: Tipo di posizionamento box nella grafica.

Il *PosMode* 16 disattiva la freccia di spostamento in altezza. In fase di spostamento il programma riconosce il piano più alto rispetto al punto del disegno su cui si sta lavorando. E' possibile spostarsi su piani posizionati ad altezze differenti in fase di movimento utilizzando i tasti freccia su e freccia

giù della tastiera.

> ! Utilizzando questo tipo di posizionamento gli snap e le collisioni vengono disattivati automaticamente. Se si necessita di precisione nell'inserimento degli oggetti si consiglia di non utilizzare questa modalità.

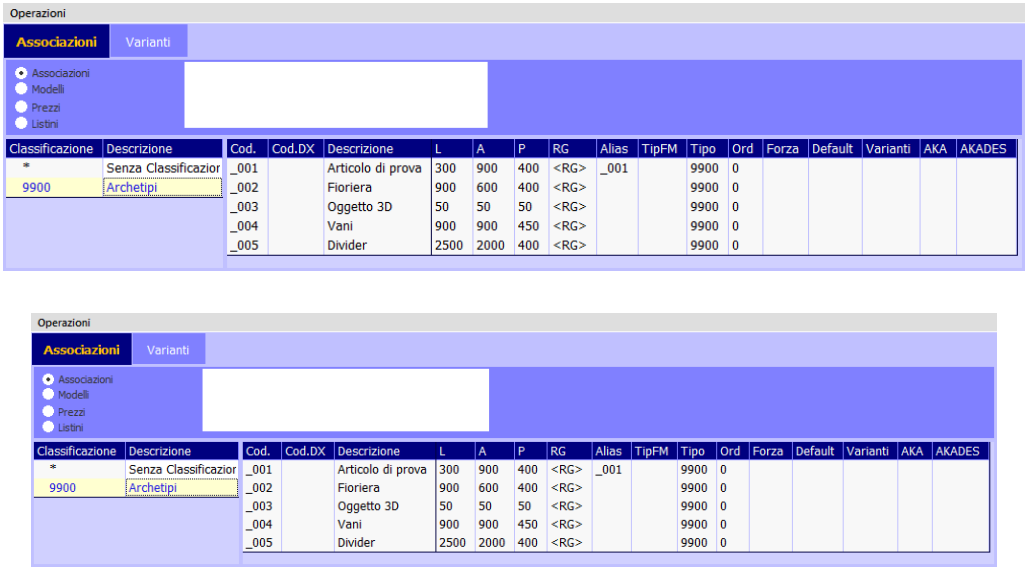


Figura 4.5: Causali fuori misura.

4.2 Varianti della regola

Il cuore della definizione della regola è l'elenco delle varianti dichiarate, visibile nella parte inferiore della schermata dell'editor. Ogni variante è impostata in una riga di una griglia le cui colonne, hanno il seguente significato:

- Variante: bisogna digitare il nome della variante desiderata, oppure si ricerca nell'apposita tabella a cui si accede con i tasti Shift+?;
- Il tasto Varianti, situato nella barra degli strumenti, consente di entrare nell'editor della variante;
- Tipo: spostandosi nella seconda colonna e cliccando sulla freccia a destra, si aprirà una casella di riepilogo contenente vari tipi , vediamone il significato:
- Alias: si utilizza questa funzionalità quando c'è la necessità di sdoppiare una variante, ossia leggerla da una tabella con un nome e utilizzarla con un altro. L'alias è gestito anche nell'anagrafica delle varianti e in quel caso è comune a tutte le regole;
- Validità: in questa colonna possiamo condizionare la presenza di una variante al verificarsi di un determinato evento; La validità è una condizione che abilita o disabilita l'intera variante e normalmente è condizionata dalle varianti precedenti;

- Descrizione: permette di dare una descrizione alle varianti inserite localmente. Con la barra di scorrimento si visualizzano le altre colonne disponibili.
- Col.Codice: Qui è possibile inserire una formula di validità non legata all'intera variante ma a ogni opzione della stessa. L'opzione è valida se almeno la formula ha risultato positivo. Attenzione, almeno una opzione deve essere valida;
- Comando: viene utilizzato principalmente all'interno della regola TESTA.erg (regola di testata) per eseguire dei comandi particolari legati al tipo di variante selezione da tabella o selezione SQL. Può essere utile usarlo anche in altre regole del catalogo o nel menu elaborazioni [5];
- Valori: Viene utilizzato nel caso di varianti di tipo libero per fornire all'utente una scelta di valori. Questi valori valgono solo per la regola in questione;
- Minimo/Massimo: nel caso si voglia controllare che una determinata variante di tipo misura non assuma valori al di fuori di un minimo e un massimo consentiti;
- C1: è possibile personalizzare lo sfondo dove viene descritta la variante scegliendo la tonalità tra una gamma di 9 colori predefiniti (0-9);
- C2: è possibile descrivere la variante con 3 tipi di caratteri diversi rispetto allo standard (0-3).

Tipi di variante

Tipo	Descrizione
Variante	è il tipo più frequente in quanto si riferisce alle varianti classiche, si seleziona automaticamente scegliendo una variante dall'elenco visualizzato con ?, va invece selezionato quando si scrive manualmente il nome della variante.
Libero	è molto usato per inserire delle varianti al volo, senza dover passare per la tradizionale gestione. I valori vanno inseriti nell'omonima colonna usando l'editor di testo messo a disposizione (ogni valore seguito da vai a capo), oppure possono essere digitati liberamente dall'operatore nel momento in cui viene caricata la regola.
Valore Fisso	è simile alla variante di tipo libero, ma gestisce un solo valore fisso non modificabile dall'utente.
Sagoma	si mette a disposizione dell'utente un editor per la costruzione di sagome libere simile ad un CAD bidimensionale.
Gruppo	nel *GRUPPI.INI* del catalogo possono essere dichiarati alcuni gruppi di varianti che trovano utilizzo in più regole. In tali regole basterà richiamare il gruppo e automaticamente verranno visualizzate tutte le varianti che lo compongono.

[5] Sconsigliamo l'utilizzo nelle regole normali.

Tabella 4.2 : continua nella prossima pagina

Tabella 4.2 : continua dalla pagina precedente

Tipo	Descrizione
Aggiunta	ha lo stesso funzionamento di un gruppo per quanto riguarda la definizione, ma serve a gestire le regole speciali ossia quelle con aggiunte e con divisioni.
Sel.SQL	viene generalmente usata nella regola di testata (testa.erg), che al momento non abbiamo ancora descritto. E' necessaria per estrarre dei dati dalle tabelle anagrafiche. Nella colonna comando viene inserito il nome del comando, mentre l'elaborazione viene descritta nel gruppo relativo a quel comando nel valutatore della regola.
Selezione da tabella	anche questa viene generalmente usata nella regola di testata. Consente di estrarre dei dati dalle tabelle di evento.
Sel. Multiple	consente di selezionare più opzioni.
Misura	è come la variante di tipo libero ma accetta solo valori numerici.
NoSi e SiNo	sono varianti utilizzabili come switch che restituiscono valore 0 o 1 . La prima avrà come default No, la seconda Sì.
Separatore	in presenza di un elevato numero di varianti, possiamo creare dei raggruppamenti titolati, che permettano una elencazione più chiara. In testa alle varianti che faranno parte del capitolo, scriviamo il nome dello stesso nella colonna Varianti e gli attribuiamo il tipo separatore. Nel momento in cui si vanno a definire le opzioni, in collaudo o in costruzione dell'ordine, prima di accedere alle varianti, dobbiamo aprire il relativo capitolo.
Nascosta	nasconde la variante nella fase di collaudo o costruzione. Viene utilizzata quando all'interno della regola viene fatta l'assegnazione di una variante con il comando *LETV*. Vedremo di approfondire l'argomento in sede di spiegazione del linguaggio delle regole.
Colore	in fase di costruzione viene aperta la finestra di scelta dei colori di *Windows*, il colore selezionato viene memorizzato come valore numerico nella variante.
Font	viene aperta la finestra *Windows* di scelta del tipo di carattere. Le informazioni relative al tratto, dimensioni e colore vengono memorizzate nella variante.
File	in fase di collaudo o costruzione viene aperta la finestra per la ricerca dei file, il file selezionato viene salvato con tutto il percorso nella variante.

Tabella 4.2 : continua nella prossima pagina

Tabella 4.2 : continua dalla pagina precedente

Tipo	Descrizione
Immagine	viene aperta una finestra per la ricerca delle immagini, vengono visualizzati solo i file di tipo jpg o bmp. Una volta selezionata, l'immagine può essere modificata. Viene salvata nella variante. Editor in fase di collaudo o costruzione viene aperto un editor di testo, il contenuto viene salvato nella variante.
Default	consente di assegnare delle costanti nella regola utilizzando una tabella variante. Praticamente, se ad una variante viene associato il tipo default, tutte le opzioni di tale tabella vengono assegnate come variabili della regola
CatGem	il lettore può trascurare questo tipo, in quanto viene usato per particolari collegamenti gestionali.

Tabella 4.2: Tipi di varianti.

4.3 Gruppi di valutazione

Nella parte alta della regola sono definiti i gruppi di valutazione, ossia i vari script che sono elaborati in risposta alle varianti per ottenere un risultato.

Nel linguaggio delle regole applicato a ***3CadEvolution*** , i gruppi di valutazione sono solo 3:

1. Grafica: è il gruppo principale, serve a disegnare l'oggetto e creare ogni riferimento (distinta base, produzione, lavorazioni ect.).
2. Eccezioni: serve a mandare dei messaggi per informare l'utente che una determinata combinazione di varianti non è valida: in realtà il gruppo eccezioni non dovrebbe neanche esistere su una regola ben scritta, per quanto possa servire in fase di collaudo.
3. Extra: serve per operazioni particolari come lo spostamento degli oggetti sui muri ad angolo.

4.4 Sintassi delle regole

Nel precedente capitolo abbiamo descritto come vanno impostate le regole, quali sono i dati preliminari da definire e, attraverso un esempio, abbiamo anche potuto vedere come si sviluppa il codice di una regola. In questa sede cercheremo di spiegare tutti i simboli, le funzioni ed i comandi disponibili nel linguaggio di ***3CadEvolution*** .

Definizione di riga

Con la parola riga si intende una qualsiasi sequenza di caratteri che rappresenti un'espressione valida per il valutatore delle regole. Per definizione una riga deve iniziare a capo rispetto a qualsiasi scrittura precedente che non le appartenga, ovvero a sinistra del primo carattere di una riga non può comparire nulla se non elementi di tabulazione.

Definizione di gruppo

Si definisce gruppo una sezione di un file di regola formata da un'intestazione e da una o più righe. Un gruppo abbraccia tutto quello che è compreso tra un'intestazione (inclusa) e l'intestazione successiva (esclusa) oppure tra un'intestazione (inclusa) e la fine del file, nel caso si tratti dell'ultimo gruppo definito. L'intestazione è composta da un nome racchiuso tra parentesi quadre.

I gruppi vengono visualizzati nel formato *ASCII* della regola. L'operatore più esperto potrà usare l'editazione del file *ASCII* quando ad esempio dovrà copiare righe di codice da una regola all'altra.

La definizione di gruppo presenta un nome racchiuso tra parentesi quadre e una o più righe a seguire.

Nel codice 52 viene riportato un gruppo denominato [INFO] che si trova sempre all'interno del file *ASCII* di una regola poiché ne definisce le caratteristiche principali (impostate dall'operatore nella scheda Definizioni della Regola).

Codice 52. Gruppo [INFO].

```
[INFO]
Descrizione =
Commenti =
Spostabile =
Flagnuovo =1
Dim.minima =0,0,0
Dim.massima=5000,5000,5000
Dim.default=1000,1000,1000
```

Nel codice 53 viene riportato un altro importante gruppo che si trova in tutte le regole che utilizzano componenti le cui caratteristiche sono definite da tabelle di varianti; il gruppo *GENE* riporta il tipo di variante ed il nome della relativa tabella (impostate manualmente nella scheda Definizione della Regola).

Codice 53. Gruppo [GENE].

```
[GENE]
R,SQLCLI,,,Selezione Cliente x
S,SP,,,Spessore
T,TIPOART
R,SQLMATE,,,Scelta Materiale x
T,RIVESTIM,RIVEST,$(TIPO:0)=1,Rivestimento Esterno
```

```
T,RIVESTIM,RIVINT,$(TIPO:0)=1,Rivestimento Interno
```

Nel codice 54 sono riportati alcuni dei gruppi che si trovano in tutte le regole, a cui si accede scegliendo una voce dalla casella Gruppo nella scheda Valuta di una Regola.

Codice 54. Gruppi [PREV], [CODICE], [GRAFICA].

```
di una Regola:
[PREV]
// ---------------- Gestione DB ---------------------
// prova
#LET l=$larg:0
#LET a=$alt:0

[CODICE]
#LETS codicep=$(sqlcli:0)
#LETS codicep=$(CODICEp)$[format;$(larg:0);0000]\
$codicep,$desmate

[GRAFICA]
cerchio 0,0,300,300,&HFF8000,12
wmf C:\logo\Idea.bmp,0,0,100,200
```

4.4.1 Caratteri riservati

La sintassi delle regole di ***3CadEvolution*** prevede l'utilizzo di un insieme di caratteri riservati. Questi sono:

CARATTERI RISERVATI

Simbolo	Funzionalità
//	Commento oppure ## oppure ‘ (apice)
{ }	Formula (parentesi = codice ascii 123,125)
( )	Priorità nelle formule o valore di una variante
[]	Priorità nelle funzioni
+	Operatore di addizione
-	Operatore di sottrazione
/	Operatore di divisione
&	Operatore AND logico
|	Operatore OR logico
=	Operatore di uguaglianza tra numeri
!	Operatore 'diverso' tra numeri

Tabella 4.3 : continua nella prossima pagina

Tabella 4.3 : continua dalla pagina precedente

Simbolo	Funzionalità
!!	Operatore 'diverso' tra stringhe
==	Operatore di uguaglianza tra stringhe
<	Operatore minore tra numeri
<=	Operatore minore-uguale tra numeri
<<	Operatore minore tra stringhe
>	Operatore maggiore tra numeri
>=	Operatore maggiore-uguale tra numeri
>>	Operatore maggiore tra stringhe
:	Prefisso per l'indice di colonna in una tabella
,	Separatore di colonna nelle tabelle
$	Prefisso di variabile o di variante
\	Continuazione della riga attuale nella successiva (a capo)
#	Prefisso per i comandi
*	Annullamento di riga in caso di risultato nullo od operatore di moltiplicazione

Tabella 4.3: Caratteri riservati.

Caratteri // oppure ## - Commento

Consente di inserire parole di commento all'interno dei file delle regole. Tutto quello che compare alla destra dei simboli // fino al termine della riga, viene ignorato dal valutatore delle regole.

Esempio 1. Utilizzo caratteri //(1)
Il valutatore interpreta l'istruzione che è la parte di riga posta a sinistra dei caratteri //, mentre ignora la parte a destra, in questo caso Larghezza non standard .

Codice 55. Inserimento di un commento.

```
1 #IF $l>300 // Larghezza non Standard
```

Esempio 2. Utilizzo caratteri //(2)
Il valutatore in questo caso ignora completamente la riga.

```
// Gestione del Database
```

Carattere \(barra retroversa) - continuazione di una riga nella successiva

Spesso può accadere che la lunghezza delle righe in una regola ecceda le dimensioni dello schermo, con conseguente riduzione della leggibilità della stessa. L'uso della barra retroversa a termine riga informa il valutatore che la continuazione della linea viene rimandata alla riga successiva, dove la medesima situazione può verificarsi a sua volta.

Esempio 3. Utilizzo carattere \.

Codice 56. Andare a capo.

```
001,l=100,p=200,a=300,c1=1,c2=1\
            ,sl=200,sa=200
```

Carattere * (asterisco) - elisione di riga in caso di risultato nullo

Il carattere asterisco (*), che va sempre posto tra la parentesi tonda aperta e il nome di una variabile, impone al valutatore di annullare completamente la riga, nella quale questa scrittura è presente, nel caso il valore restituito dalla variabile corrisponda alla stringa nulla. Questa notazione è utile nei casi in cui una porzione della stringa da generare sia assolutamente necessaria al significato della stessa e in mancanza della quale è di rigore non produrre nulla.

Consente, ad esempio, di evitare la produzione di quei codici in cui uno o più componenti elementari risultano indefiniti. Generalmente questa opzione viene utilizzata solo nell'ambito della generazione di codici, cosa che non esclude la sua applicazione anche in altre situazioni come, ad esempio, la grafica.

Esempio 4. Utilizzo carattere *.
Con il comando *LETS* si assegna ad una variabile un valore, in questo caso la variabile COD assume significato solo se la variante FAB in colonna tre non è vuota.

Codice 57. Troncamento della riga in caso la variabile non sia definita.

```
#LETS COD=BXL/$(*FAB:3)
```

Caratteri { } - formula

Calcola numericamente l'espressione contenuta tra parentesi e ne converte il risultato in stringa. L'espressione deve essere sintatticamente corretta, ovvero concorde con la tradizionale notazione per le espressioni matematiche. In caso di errore di valutazione viene ritornata la stringa vuota.

Esempio 5. Utilizzo caratteri { }.
Supponendo che la variabile $(vani) contenga il valore 2, il codice 58 verrà interpretato come:
Vano 4 Cassetti

Codice 58. Conversione di un calcolo algebrico in una stringa.

```
1 Vano {$(vani)*2} Cassetti
```

Caratteri ( ) - parentesi tonde

(1) Priorità nelle formule. L'uso delle parentesi tonde per impostare la priorità di valutazione nelle formule è identico a quello della notazione matematica classica, tipico dei moderni linguaggi di programmazione.

(2) Valore di una variante. Un altro tipo di sintassi che vede coinvolte le parentesi tonde riguarda il riferimento agli elementi delle tabelle delle varianti (vedi cap. 1)

Carattere , (virgola) - separatore di colonna

Nel file *ASCII* separa le stringhe organizzate in modo tabellare, codice 59.

Codice 59. Carattere ,(virgola) (1).

```
1 [GENE]
2 T , STR
3 T , TRAM
```

Nelle righe in cui si richiamano le tipologie, viene usata per separare tra loro i parametri, codice 60.

Codice 60. Carattere ,(virgola) (2).

```
1 8,SP=$P-30,P=30 // zoccolo
```

Carattere $ (dollaro) - prefisso di variabile

Il carattere $ serve per fare riferimento ad una variabile e deve precedere il nome della variabile nella stesura della regola. Vi sono due tipi di variabili, quelle definite dall'utente e quelle che fanno riferimento agli elementi di una tabella varianti. La sintassi da seguire per l'utilizzo delle variabili è la seguente:

La sintassi da usare è la seguente:

$(nomevariabile:nr.colonna)

Il numero della colonna nel caso delle varianti definite dall'utente è zero. Per le varianti assegnate con il comando *LET*:

La sintassi da usare è la seguente:

$(nomevariabile)

Il carattere $ può essere usato anche in un contesto particolare, nel senso che genera una variabile fittizia, secondo questa sintassi:

La sintassi da usare è la seguente:

$M:N

dove M ed N sono dei numeri interi, con M minore di N; quello che si ottiene è una stringa che elenca tutti gli interi da M a N, separati da un [;] .

Esempio 6. Comando $ per generare una stringa di numeri.
Il codice 61 genera la stringa:
1;2;3;4;5;6;7;8;9;10

Codice 61. Carattere $.

```
1 $1:10
```

Carattere : - riferimento agli elementi di una tabella varianti

Utilizzando l'opportuna sintassi, come spiegato nel paragrafo precedente, il carattere : restituisce la voce presente nella colonna indicata dal carattere che segue i :, in corrispondenza della riga selezionata in quel momento, rispettivamente ad una certa tabella di Variante. Deve essere specificato naturalmente il nome della tabella variante di cui si vuole ottenere lo specifico valore. Facciamo presente al lettore che per le varianti tabellate la numerazione delle colonne parte da uno, mentre per le varianti di tipo libero c'è un'unica colonna, la zero.

Codice 62. Selezionare l'elemento colonna di una variante.

```
1 $(COL:1)
```

Carattere # (diesis o cancelletto) - prefisso dei comandi

Il carattere diesis viene utilizzato all'interno delle regole come prefisso dei comandi.

Carattere ! (punto esclamativo) - prefisso di funzioni

Il punto esclamativo ad inizio riga è il prefisso per le funzioni.

Caratteri = , ! , < , > - confronto

=: uguale

!: diverso

<: minore

>: maggiore

Caratteri & , | - operatori logici

Gli operatori logici AND e OR vengono usati generalmente insieme al comando *#IF* (diramazione condizionata), oppure per calcolare il valore true o false di un'espressione racchiusa fra parentesi graffe. Approfondiremo quest'ultimo concetto nel prossimo paragrafo.

4.4.2 La calcolatrice di *3CadEvolution*

In ***3CadEvolution*** troviamo un calcolatore algebrico, vale a dire uno strumento che permette di calcolare il valore di una espressione algebrica, scritta come stringa. Il calcolatore esegue la valutazione purché l'espressione sia racchiusa fra parentesi graffe. Le operazioni e gli operatori ammessi da questo calcolatore sono elencati di seguito, alcuni di essi li abbiamo già citati precedentemente.

Funzioni numeriche

Funzione	Descrizione
ABS	restituisce un intero positivo corrispondente alla parte intera del numero rappresentato dalla stringa (valore assoluto)
SGN	restituisce 1 se il numero rappresentato dalla stringa è positivo, -1 se negativo, 0 se è lo zero
SIN	restituisce il valore del seno di un angolo espresso in gradi
SND	simile a SIN ma restituisce un risultato più preciso
COS	restituisce il valore del coseno di un angolo espresso in gradi
CSD	simile a COS ma restituisce un risultato più preciso
TAN	restituisce il valore della tangente di un angolo espresso in gradi
TND	simile a TAN ma restituisce un risultato più preciso
ASN	restituisce il valore dell'arcoseno, espresso in gradi
ACS	restituisce il valore dell'arcocoseno, espresso in gradi
ATN	restituisce il valore dell'arcotangente, espresso in gradi
INT	restituisce la parte intera di un numero o espressione
RND	restituisce il valore arrotondato di un numero decimale (.5)
RN1	restituisce 1 cifra decimale dopo la virgola
RN2	restituisce 2 cifre decimali dopo la virgola
EXP	Restituisce il valore dell'esponenziale di base E
LOG	restituisce il valore del logaritmo in base E
SQR	restituisce il valore della radice quadrata di un numero

Tabella 4.4 : continua nella prossima pagina

Tabella 4.4 : continua dalla pagina precedente

Funzione	Descrizione
NOT	dà come risultato 1 se il valore inserito è 0 e restituisce 0 in tutti gli altri casi

Tabella 4.4: Funzioni numeriche della calcolatrice.

Esempio 7. Assegnazione di un valore numerico ad una variabile.
In A sarà contenuto il valore 10.

```
#LET A={5+5}
```

Esempio 8. Assegnazione di un valore restituito da una funzione numerica.
In A sarà contenuto il risultato di tutta l'espressione.

```
#LET A={LOG(3)*SIN(30)/SQR(2)}
```

4.4.3 Operatori logici

&: corrisponde all'operatore logico AND

|: corrisponde all'operatore logico OR

Esempio 9. Utilizzo operatore logico &.
In P sarà contenuto il valore –1 perché sono due condizioni vere (ciascuna vale –1).

Codice 63. Confronto tra due espressioni con operatore logico AND.

```
#LET A=3
#LET P={($A<5) & ($A>1)}
```

4.4.4 Confronto di stringhe

== per stabilire se le due stringhe sono uguali

!! per stabilire se le due stringhe sono diverse

>> per stabilire se la prima stringa è maggiore della seconda, in base all'ordine dei codici *ASCII*

<< per stabilire se la prima stringa è minore della seconda, in base all'ordine dei codici *ASCII*

Esempio 10. Confronto tra due stringhe, se la prima è minore della seconda. In P sarà contenuto il valore –1 (cioè true) perché il codice ASCII di B è minore di quello di a.

Codice 64. Confronto tra due stringhe con << .

```
1 #LET P={BBB<<aaa}
```

Esempio 11. Confronto tra due stringhe, se sono uguali.
In P sarà contenuto il valore 0 (cioè false).

Codice 65. Confronto tra due stringhe con == .

```
1 #LET P={aab==aaa}
```

4.4.5 Operatori numerici

>= controlla se il primo valore è maggiore o uguale del secondo

<= controlla se il primo valore è minore o uguale del secondo

<>, !=, ! controlla se i due valori sono diversi

= controlla se i due valori sono uguali

> controlla se il primo valore è maggiore del secondo

< controlla se il primo valore è minore del secondo

\+ esegue la somma dei due valori

\- esegue la differenza di due valori

* esegue il prodotto di due valori

/ esegue la divisione di due valori

Esempio 12. Divisione.
Assegna ad A il risultato 0.625.

Codice 66. Divisione di due valori (1).

```
#LET A=5/8
```

Codice 67. Divisione (2).

```
#LET A={(16+74)/(15*2)-1}
```

assegna ad A il risultato 2.

Il calcolatore è in grado, inoltre, di stabilire se un elemento appartiene a un insieme oppure no. Verrà restituito il valore -1 nel caso l'elemento appartenga all'insieme, 0 in caso contrario.

La sintassi da usare è la seguente:

---

el[el1; el2; el3; ...; elN]

Codice 68. Appartenenza di un elemento ad un insieme (1).

```
#LETS PIPPO={aa[1, 2, aa, 3, 4, 5]}
```

In PIPPO sarà contenuto il valore –1 poiché l'elemento aa appartiene all'insieme indicato.

Codice 69. Appartenenza di un elemento ad un insieme (2).

```
#LETS PIPPO={a[1 , 2, aa, 3, 4, 5]}
```

In PIPPO sarà contenuto il valore 0 poiché l'elemento a non appartiene all'insieme indicato.

Il calcolatore permette, dato un intervallo di numeri interi, di ottenere ad ogni valutazione della regola, un valore diverso compreso nell'intervallo definito.

La sintassi da usare è la seguente:

$(RN0n1n2)

I numeri n1 e n2 devono avere il formato 000

Esempio 13. Generare un numero random.

Codice 70. Comando RN0n1n2.

```
#LETS PIPPO=$(RN0001200)
```

La variabile PIPPO ritornerà ad ogni valutazione un valore casuale compreso tra 1 e 200.

Esiste una funzione che controlla la presenza di una determinata variante e restituisce il valore 1 nel caso esista, 0 se la variante non è presente.

La sintassi da usare è la seguente:

$(<nomevariante>_SPECIFIED)

Codice 71. Presenza di una determinata variante.

```
#LETS PIPPO=$(STR_SPECIFIED)
```

La variabile PIPPO assumerà il valore 1 se la variante STR esiste, 0 nel caso contrario.

4.5 Variabili di sistema di utilità

Elenchiamo di seguito le variabili di sistema disponibili in ***3CadEvolution*** . Queste variabili si distinguono in quanto iniziano con il carattere '.' . Esse restituiscono informazioni di vario genere, non strettamente legate alle regole grafiche, ma di utilità ad esempio in alcuni test o a livello di stampe e collegamenti gestionali.

Variabili di sistema

Variabile	Descrizione
.COMPUTERNAME	restituisce il nome del computer
.DATAORA	restituisce la data e l'ora nel formato gg/mm/aaaa-hh.mm
.ORA	ora corrente hh.mm
.PERCORSO	restituisce il nome del catalogo corrente
.SYSDIR	restituisce il nome della cartella di *Windows* system

Tabella 4.5 : continua nella prossima pagina

Tabella 4.5 : continua dalla pagina precedente

Variabile	Descrizione
.TEMPDIR	restituisce la cartella dove vengono salvati i file temporanei
.TEMPFILE	restituisce il nome del file temporaneo
.USERNAME	mostra il nome dell'utente di rete con cui è stato fatto il login
.WINDIR	restituisce il nome della cartella di *Windows*
.RIVENDITORE	restituisce 1 se il programma è in modalità ridotta altrimenti 0
.INLINGUA	dichiara la lingua corrente
.DESKTOP	cartella desktop
.CURDIR	cartella di lavoro del programma
.USERNAME	nome utente
.TEMPFILE	nome file temporaneo
.DATA	data
.TMKEY	numero di serie della chiave hardware in uso
.VERSIONE	versione del programma ultima cifra (es. 601)
.MODULOLITE[n]	indica se abilitato il modulo lite nr.[n]
.MODULO[n]	indica se la chiave contiene il modulo [n]
.GLOBALE[n]	elenco di variabili globali del programma in lettura e scrittura, queste variabili non vengono mai azzerate nel funzionamento del programma pertanto consentono di mantenere contatori globali ecc. [n] vale da 1 a 99. si tratta delle sole variabili di sistema che è possibile modificare!
.ISECADPRO	ritorna 1 se versione pro altrimenti 0
.PERCORSOPRO	solo versione pro indica il percorso del programma
.PERCORSOLITE	solo versione lite indica il percorso del programma
.GRUPPOUTENTE	solo versione lite indica a che gruppo di utenti corrisponde l'utilizzatore
.RND.[n1].[n2]	genera un numero casuale tra n1 e n2

Tabella 4.5 : continua nella prossima pagina

Tabella 4.5 : continua dalla pagina precedente

Variabile	Descrizione

Tabella 4.5: Variabili di sistema.

4.6 Comandi del linguaggio delle regole

4.6.1 Comando *#LET*

Il comando *#LET* consente di assegnare una quantità ad una variabile a singolo valore quali le variabili di sistema o quelle utente; non è consentito l'uso di *#LET* con le variabili tabella in quanto i loro valori sono codificati in modo statico direttamente nel database. A differenza dell'usuale notazione per le variabili, nel comando *#LET* ci si riferisce alla variabile semplicemente utilizzando il suo nome privato delle parentesi e del prefisso $. E' importante osservare che si possono fare più assegnazioni con il comando *#LET*, purché tutte sulla stessa linea.

Esiste la possibilità di assegnare in una regola delle variabili il cui valore rimane associato al box, queste devono chiamarsi *LOC_BXn* dove n è un valore da 0 a 10. La gestione di questo tipo di variabile è utile, ad esempio, nel caso in cui lo stesso frontale possa essere inserito in diverse posizioni, in base al tipo di elemento finito che si vuole costruire (colonna a terra o colonna sovrapposta). Se nella regola della colonna a terra assegniamo alla variabile *LOC_BX0* il valore 1 e nella regola della colonna sovrapposta, alla stessa variabile diamo il valore 2, nella regola che gestisce i frontali o i ripiani possiamo gestire i diversi punti di inserimento, controllando il valore assunto di volta in volta dalla variabile *LOC_BX0*.

Anche per questo tipo di assegnazione si deve usare il comando *#LET* (o *#LETS*la sezione successiva).

Esempio 14. Assegnare un valore alla variabile.

Codice 72. Comando *#LET* (1).

```
1   #LET A=625 // imposto l'altezza a 625 millimetri
```

In questo esempio viene assegnato il valore 625 alla variabile $A che identifica l'altezza.

Codice 73. Comando *#LET* (2).

```
1   #LET B=5+5
```

In questo esempio viene assegnato il valore 10 alla variabile B.

Codice 74. Comando *#LET* (3).

```
#LET A=5, B=7
```

In questo esempio viene assegnato il valore 5 alla variabile A e il valore 7 alla B, con un unico comando.

4.6.2 Comando *#LETS*

Il comando *#LETS* consente di assegnare ad una variabile un valore di tipo stringa. A differenza dell'usuale notazione per le variabili, nel comando *#LETS* ci si riferisce alla variabile semplicemente utilizzando il suo nome privato delle parentesi e del prefisso $.

Esempio 15. Assegnare ad una variabile una stringa.

Codice 75. Comando *#LETS* (1).

```
#LETS codice=$(sqlcli:0)
```

In questo esempio viene assegnato alla variabile codice il valore che si trova nella tabella sqlcli nella colonna 0.

Codice 76. Comando *#LETS* (2).

```
#LETS prova=primo, secondo, terzo, quarto, quinto
```

Qui invece la variabile è definita dall'utente e corrisponde a $(prova); il valutatore gli assegna il valore primo, secondo, terzo, quarto, quinto; se si valuta $(prova:0) essa conterrà il valore primo, $(prova:1) conterrà il valore secondo e così via. Si noti l'uso della virgola come separatore.

Codice 77. Comando *#LETS* (3).

```
#LETS ALFA1=primo, ALFA2=secondo
#LET N=2
#MSG $(ALFA$(N))
```

L'ultima istruzione valuta prima il contenuto della variabile N, poi quello della variabile ALFA di indice il valore di N, quindi il valore restituito sarà quello di ALFA2, cioè secondo.

Esiste una particolarità nel comando *#LETS* con assegnazioni multiple.

Esempio 16. Assegnazioni multiple.

Codice 78. Comando *#LETS*, assegnazioni multiple.

```
#LETS alfa=primo, secondo, beta=terzo
```

Assegna alla variabile ALFA il valore primo, secondo e alla variabile beta il valore terzo.

Può essere utilizzato in alternativa il comando *#LETT* che non accetta assegnazioni multiple, in modo tale che la riga:

Codice 79. Comando *#LETT* (1).

```
   #LETT alfa=primo, secondo, beta=terzo
```

Corrisponde all'assegnazione alla variabile ALFA del valore primo, secondo, beta=terzo. Questo comando è indispensabile nella preparazione delle stringhe per SQL nelle elaborazioni.

Codice 80. Comando *#LETT* (2).

```
   #LETT sql = $(sql) AND left((RORDINE.CODART),2) >= \
               '$(mod1:0)' AND. .
```

4.6.3 Comando *#LETV*

Il comando *#LETV* consente di assegnare un valore alla prima colonna di una variante tabellata. Automaticamente naturalmente vengono assegnate anche le altre colonne relative all'opzione identificata dal valore assegnato.

Codice 81. Comando *#LETV*.

```
   #LETV str=02
```

In questo caso è come se assegnassimo il valore 02 alla prima colonna della variante *"STR"*. La procedura restituirà tutti i valori associati a quella opzione che naturalmente deve esistere nella variante stessa:

STR=02,02,noce,1

4.6.4 Comando *#LETS_MACROFINALE*

Il comando *#LETS_MACROFINALE*, permette di eseguire una macro alla fine di un caricamento della regola.

La sintassi è: *#LETS_MACROFINALE*=nome_macro .

Esempio 17. Regola dove si utilizza il divider ***3CadEvolution*** esegue per prima:

Codice 82. Comando *#LETS_MACROFINALE*.

```
   #LETS _MACROFINALE=FIANCHI

// ZOCCOLO
#IF $(VZOC:1)==1 | $(ZOC)=1
  205, SL=0, SA=0, L=$l, SP=$P-$(SPE_ZOC)-$(SPE_FRN) //
     ZOCCOLO
#ENDIF

// BASE - CAPPELLO - CORNICE
// Cappello Parete
203.001, SL=0, SA=$A-$(SPE_cap)-$(HCOR), SP=20*$(intbase), L=
   $L, p=$p-$(spe_frn)-$(aria)
// Base Interparete
203.002, SL=0, SP=20*$(intbase), L=$L, P=$P-$(spe_frn)-$(aria
   ), sa=$(HZOC)
```

Poi esegue la macro dei vani (per graficare i frontali ddivani), poi alla fine passa per macro Fianchi.

La macro dei fianchi non serve venga messa nella regola, viene richiamata in automatico come quelle per divider e delle aggiunte.

Questo serve ad esempio se nei fianchi bisogna tener conto, per il codice, del tipo di frontale. Se non utilizzavo la MACROFINALE prima grafica i fianchi e poi i frontali, in questo caso non è possibile, perché quando grafico il fianco non so che frontale viene applicato.

Se utilizzo questo comando nella macro dei vani carico una variabile che posso testare e utilizzare nella macro dei fianchi.

4.6.5 Comandi *#IF #ELSEIF #ENDIF*

Il comando *#IF* permette di effettuare dei controlli all'interno della regola, in base ai quali svolgere poi operazioni tra loro diverse. Un esempio testuale può essere: se la larghezza > 900 disegna due ante, altrimenti disegna un'anta. Il controllo (scritto in grassetto) può essere composto da più test di verifica, ed in generale viene definito condizione logica.

Lo schema di una struttura di controllo *#IF-#ELSEIF-#ELSE-#ENDIF* è il seguente:

Codice 83. Comando *#IF-#ELSEIF-#ELSE-#ENDIF* (1).

```
   #IF condizione logica // SE condizione logica
```

```
  righe
#ELSEIF condizione logica //ALTRIMENTI SE condizione logica
  righe
#ELSE // ALTRIMENTI
  righe
#ENDIF // FINE
```

Tale struttura consente di richiamare sequenze di comandi (righe) in accordo al risultato di diverse condizioni logiche. Il valutatore verifica se il risultato della condizione logica è vero, nel qual caso esamina le righe immediatamente successive alla parola chiave *#IF*; in caso contrario esamina le condizioni logiche (una alla volta) associate alle parole chiave *#ELSEIF* (se presenti) comportandosi in modo analogo. Se le condizioni logiche risultano essere tutte false, il valutatore esamina le righe che seguono la parola chiave *#ELSE* (se presente) fino alla parola *#ENDIF*.

Le condizioni logiche devono essere specificate di seguito (e nella stessa linea) ai comandi *#IF* o *#ELSEIF*, mentre le righe contenenti le parole chiave *#ELSE* ed *#ENDIF* non devono contenere nessun altro carattere; le righe che seguono una condizione logica o la parola chiave *#ELSE* vanno necessariamente scritte su linee separate (e sottostanti).

La forma più semplice per il comando *#IF* prevede la presenza di una condizione logica e di una riga al suo seguito mentre nella forma più complessa constatiamo un numero indefinito di annidamenti di condizioni logiche associate a parole chiave *#ELSEIF*.

Nell'esempio che segue osserviamo due strutture *#IF* annidate; il secondo *#IF* viene valutato solo se la condizione del primo risulta vera, altrimenti il controllo passa al comando #LET DA = $(TIPO:8).

Esempio 18. *#IF* annidate.

Codice 84. Comando *#IF-#ELSEIF-#ELSE-#ENDIF* (2).

```
#IF $(TIPO:1) = 1
  #IF $DA > 1280
    #LET DA = 1280
  #ENDIF
#ELSE
  #LET DA = $(TIPO:8)
#ENDIF
```

In questo secondo esempio riportiamo una struttura completa *#IF-#ELSEIF-#ELSE-#ENDIF*, nella quale (come spiegato in precedenza) una sola delle righe comprese tra le parole chiave, verrà valutata.

Esempio 19. *#IF* struttura completa.

Codice 85. Comando *#IF-#ELSEIF-#ELSE-#ENDIF* (3).

```
#IF $A > 1980
   #LET A = 1980
#ELSEIF $A < 470
   #LET A=470
#ELSE
   #LET A = $(MISURE:4)
#ENDIF
```

Infine mostriamo la forma più semplice di un comando *#IF*, nella quale la variabile P viene limitata superiormente al valore 980.

Codice 86. Comando *#IF-#ELSEIF-#ELSE-#ENDIF* (4).

```
#IF $P > 980
  #LET P = 980
#ENDIF
```

4.6.6 Comando *#IF*

Facciamo notare al lettore, che in alcuni casi è possibile usare solo il comando *#IF* senza il relativo *#ENDIF*.

La sintassi da usare è la seguente:

#IF <condizione> <istruzione per condizione vera> <istruzione per condizione falsa>

Esempio 20. *#IF* senza il relativo *#ENDIF*.

Codice 87. Comando *#IF*.

```
#IF $a>1000 #MSG A superiore a 1000 #MSG A inferiore a 1000
```

4.6.7 Comandi *#IFDEF #IFNDEF*

Il comando *#IFDEF* (IF DEFined) presenta il medesimo funzionamento del comando *#IF* nella sua forma più semplice, ovvero *#IF-#ELSE-#ENDIF*, con la differenza che l'espressione logica ad esso associata viene valutata soltanto dal punto di vista della consistenza: se l'espressione riportata di seguito alla parola chiave *#IFDEF* è definita, verranno valutate le righe comprese tra *#IFDEF* ed *#ELSE*, altrimenti l'attenzione viene spostata sulle righe comprese tra *#ELSE* ed *#ENDIF*.

Vediamo di chiarire il concetto di definizione di una variabile; una variabile si dice definita se è possibile reperire il suo valore corrente. Una variabile che si riferisce ad una tabella è definita quando esiste una riga attiva; una variabile dichiarata dall'utente è definita solo se le è stato assegnato un valore.

Codice 88. Verifica se una variabile è definita, *#IFDEF*.

```
#IFDEF Espressione // SE DEFINITA Espressione
  riga
#ELSE // ALTRIMENTI
  riga
#ENDIF // FINE
```

In modo assolutamente analogo viene descritto il comportamento del comando *#IFNDEF* (IF Not DEFined) nel quale la valutazione del primo gruppo di righe è legata all'assenza di definizione per l'espressione specificata. Un'espressione è definita quando tutte le variabili al suo interno sono definite.

Il comando *#IFDEF* viene quasi sempre applicato alle variabili di riferimento delle tabelle di varianti. Applicare *#IFDEF* a variabili di sistema è un'azione priva di significato, anche se legittima perché sono sempre presenti. Esiste anche il comando *#ELSEIFDEF*.

Codice 89. Comando *#ELSEIFDEF*.

```
#IFDEF $(Colore:1)
  #LETS A=$(Colore:1)
#ELSEIFDEF $(Scocca:1)
  #LETS A=$(Scocca:1)
#ELSE
  #LETS A=BIANCO
#ENDIF
```

Nell'esempio appena riportato andiamo prima a verificare se $(Colore:1) è reperibile oppure no, e cioè se è stata attivata la tabella Colore, poi si verifica la stessa cosa anche sulla tabella scocca. L'istruzione che segue *#IFDEF* viene eseguita se $(Colore:1)=VALORE VALIDO, e cioè se esiste almeno un opzione attiva della tabella (qualsiasi sia il contenuto di $(Colore:1)), altrimenti si passa al controllo sulla variante scocca. Se questa è definita viene eseguita l'assegnazione che segue, altrimenti si passa alle istruzioni che seguono *#ELSE*.

Senza variazioni di sorta si sarebbe potuto sostituire la condizione $(Colore:1) con $(Colore:1)=1, che dal punto di vista del risultato è praticamente equivalente.

4.6.8 Comando *#FOR*

La sintassi da usare è la seguente:

#FOR <valori sep. ;>, [Condizione], [Variabile ciclo] righe #ENDFOR

Il comando *#FOR* consente di scrivere regole più compatte, nel senso che la ridondanza introdotta dalla programmazione esclusivamente sequenziale può essere eliminata. Ad esempio, un blocco condizionale *#IF-#ELSEIF* le cui condizioni si concentrino su un'unica variabile può spesso essere sostituito dal comando *#FOR* opportunamente parametrizzato.

Il funzionamento viene riassunto nel seguente modo: ad una variabile di sistema, di default $(i) o altrimenti specificata nel parametro [variabile ciclo], viene assegnato il primo dei valori dopo la parola chiave *#FOR*.

Di seguito viene valutata la condizione logica posta al termine della riga contenente l'istruzione *#FOR*. Se la condizione è falsa si inizia la valutazione delle righe comprese tra *#FOR* ed *#ENDFOR*, al cui termine viene assegnato il valore successivo alla variabile ciclo ed il ciclo si ripete tante volte quanti sono i valori riportati dopo il v. Altrimenti, se la condizione logica è vera, si passa direttamente alla riassegnazione della variabile ciclo (senza valutare le righe) utilizzando il valore successivo nella lista dei valori e la condizione logica viene valutata nuovamente.

Intuitivamente si sarebbe portati a fornire una condizione logica per la quale sia il valore True a consentire l'accesso alle righe contenute nel *#FOR*, ma il comportamento del valutatore in assenza di una possibile espressione (ricordiamo che la condizione logica è opzionale, non necessaria) è di ritornare un valore False. Quindi non includere una condizione logica nel comando *#FOR* equivale a introdurre una condizione sempre falsa e quindi il valutatore esaminerebbe sempre il contenuto delle righe per qualsiasi valore della lista.

È evidente a questo punto che la condizione logica deve necessariamente dipendere anche dalla variabile CICLO al fine di ottenere valori diversi per ogni ciclo di valutazione.

Vediamo ora un tratto di regola contenente l'istruzione *#FOR*, che non utilizza il terzo parametro, pertanto il ciclo viene sviluppato sulla variabile $(i).

Codice 90. Comando *#FOR* (1).

```
1 #FOR 0; 330; 660; 990; 1320; 1650, $(I) > $A + 220
2   005, sa=$(i) // ripiano
3 #ENDFOR
```

La variabile $(I) assume inizialmente il valore 0 e immediatamente dopo la condizione logica viene valutata: poiché il confronto *0 > altezza complessiva + 220* è sicuramente falso, la tipologia elementare 5, corrispondente al ripiano, viene richiamata e posizionata con un certo spostamento in altezza dipendente dalla variabile $(I). Si ritorna poi alla valutazione del prossimo valore per la variabile $(I).

In questo modo riusciamo a richiamare più volte la stessa riga $(I) con parametri differenti e inoltre vincoliamo l'accesso alle righe attraverso una condizione logica che deve risultare non-vera. L'istruzione *#FOR* può essere quindi parametrizzata per il caso generale (per esempio fornendo l'intera lista di quote per il posizionamento dei ripiani) e vincolata per il caso particolare (ad esempio impedendo l'inserimento di ripiani posizionati oltre l'altezza massima della colonna).

Un'altra sintassi possibile per il comando *#FOR* è la seguente:

Codice 91. Comando *#FOR* (2).

```
#FOR $1:$(ncas) //inizia da 1 fino a $(ncas)
  CASSETTO, CoL=$(variante:$(I)) //$(I) funge da contatore
#ENDFOR
```

Oppure:

Codice 92. Comando *#FOR* (3).

```
#LET ALFA=12, BETA=20
#FOR $$(ALFA):$(BETA)
  #MSG {$(I)+5}
#ENDFOR
```

L'output sarà una sequenza di dati, in colonna, che vanno dal 17 al 25.

Per finire un esempio con un ciclo FOR annidato all'interno di un altro, in questo caso è opportuno specificare una variabile di ciclo diversa da $(i).

Esempio 21. Cicli FOR annidati.

Codice 93. Cicli *#FOR* annidati.

```
#FOR $1:3, ,I // ciclo da 1 a 3 con la variabile I
  #FOR $1:2, ,j // ciclo da 1 a 2 con variabile j
    #MSG $(i)-$(j)
  #ENDFOR
#ENDFOR
```

4.6.9 Ciclo *#DO...#LOOP (#continue) (#break)*

Questa coppia di comandi crea e gestisce un ciclo.

La sintassi da usare è la seguente:

#DO [righe:]
#BREAK <condizioni di uscita> [righe:]
#LOOP <condizioni di uscita>

Dove l'esecuzione del ciclo termina quando l'istruzione di uscita diventa vera.

Codice 94. Ciclo *#DO...#LOOP (#continue) (#break)* (1).

```
#LET ALFA=0
#DO
   #LET alfa=$(alfa)+1
   #MSG $(alfa)
#LOOP $(alfa)>=5
```

La variabile ALFA viene incrementata fino a cinque, dopo il ciclo si ferma.

Codice 95. Ciclo *#DO...#LOOP (#continue) (#break)* (2).

```
#DO
   #LET alfa=$(alfa)+1
   #BREAK $a<{$(alfa)*320}
   4, SA=$(alfa)*320
#LOOP $(alfa)>=5
```

Nel codice 95 descritto abbiamo utilizzato il comando Break: in questo caso il ciclo si interrompe se l'altezza del box è inferiore al risultato dell'espressione contenuta tra le graffe.

4.6.10 Comando *#BREAK*

Serve per uscire direttamente da un ciclo DO al verificarsi di una determinata condizione, senza eseguire ulteriori istruzioni (v. esempio precedente).

4.6.11 Comando *#CONTINUE*

Serve per saltare una sequenza di istruzioni all'interno di un ciclo DO al verificarsi di una certa condizione ed arrivare direttamente alla condizione di controllo per l'uscita dal ciclo stesso.

Esempio 22. Utilizzo comando *#CONTINUE*.

Codice 96. Comando *#CONTINUE*.

```
#DO
   #LET alfa=$(alfa)+1
   #BREAK $a<{$(alfa)*320}
   #CONTINUE $(alfa)=3
   4, SA=$(alfa)*320
#LOOP $(alfa)>=5
```

In questo caso quando la variabile ALFA=3 si salta l'istruzione che segue e continua il ciclo con ALFA=4.

4.6.12 Comando *#SQL*

Questo comando permette di assegnare ad una variabile il risultato di una chiamata SQL tramite una *Query* su una tabella del database.

Codice 97. Comando *#SQL*.

```
#LETS beta=100113
#SQL alfa=SELECT rag1,ind FROM ecadmaster.dbo.clienti where
    cod ='$(beta)'
#MSG $(alfa)
```

Assegna alla variabile ALFA il risultato della *Query* SQL impostata in modo da trovare la ragione sociale e l'indirizzo del cliente 100113 nella tabella CLIENTI In questo caso si è utilizzata la variabile $(beta) per gestire la ricerca.

4.6.13 Comando *#SELECT*

Il costrutto *#SELECT* presente nel linguaggio delle regole è simile a quello presente in altri linguaggi di programmazione, in particolare è ispirato al comando *switch* del linguaggio C, C++.

La sintassi da usare è la seguente:

#SELECT [formula]
#CASE [elenco valori numerici separati da [:]]
#CASES [elenco valori stringa separati da [:]]
#DEFAULT
#ENDSELECT

Si possono mescolare #CASE e #CASES e il valutatore procede per in ordine dell'inserimento. L'istruzione #DEFAULT può essere omessa. Inoltre il costrutto *#SELECT* può essere annidato.

Esempio 23. Costrutto *#SELECT*.

Codice 98. Costrutto *#SELECT*.

```
#SELECT $(alfa)
 #CASE 1:2:3
    #lets VAR1=PROVA1
 #CASE 4:5:6
    #lets VAR1=PROVA2
 #CASES 4+4:5+5
    #lets VAR1=PROVA3 dati non valutati di ALFA
 #CASE 8:10
    #lets VAR1=PROVA3
 #DEFAULT
    #let VAR1=ALTRO
 #ENDSELECT
```

N.B. Il costrutto *#SELECT* è più efficiente(veloce) rispetto a utilizzare ripetizioni del ciclo *#IF #ELSEIF #ENDIF*.

4.6.14 Comando *#SWAP*

Questo comando permette lo scambio di valore fra due variabili.

La sintassi da usare è la seguente:

#SWAP a,b

4.6.15 Comando *#MSG*

Questo comando consente di visualizzare un messaggio. Al comando deve fare seguito il testo o la variabile che dovrà comparire a video:

Codice 99. Comando *#MSG*, che visualizza il testo.

```
#MSG Finitura non valida
```

oppure:

Codice 100. Comando *#MSG*, che visualizza il valore di una variabile.

```
#MSG Lunghezza: $(lun)
```

4.6.16 Comandi *#PUSH e #POP, e #POPALL*

Il comando *#PUSH* salva il contenuto delle variabili attive, il comando *#POP* azzera l'effetto del push. Esiste anche il comando *#POPALL* che azzera tutti gli eventuali PUSH annidati all'interno dei cicli IF.

Il comando *#PUSH* necessita di specificare il nome della variabile da salvare, mentre il comando *#POP* non ha parametri.

4.6.17 Comando *#DUMP*

Questo comando restituisce l'elenco di tutte le variabili utilizzate, in una finestra di controllo. Esiste anche la versione *#DUMPFILE* <nomefile> che salva il contenuto di tutte le variabili in un file di testo.

4.6.18 Comando *#INP*

Questo comando consente di inserire all'interno della regola un prompt con la richiesta del valore della variabile.

La sintassi da usare è la seguente:

#INP nomevar, valore di default, titolo

Codice 101. Comando *#INP* all'interno di un IF.

```
#IF $l<400
  #INP L,400, MISURA MINIMA
#ENDIF
```

Se la lunghezza dell'elemento è inferiore a 400, si aprirà una finestra con la richiesta di inserire la nuova misura, che come default proporrà 400. Questo metodo è anche molto utile per collaudare la grafica di una tipologia per diversi valori di prova.

4.6.19 Comandi *#GOTO o #JUMP o #VAI*

Questi comandi, che sono equivalenti, consentono di spostarsi in un altro programma di cui si specifica il nome.

La sintassi da usare è la seguente:

#VAI <etichetta>, [condizioni]

Esempio 24. Utilizzo comando *#GOTO*.

Codice 102. Comando *#GOTO*.

```
#LET pp=0
# alfa // etichetta ALFA
```

```
#MSG $(PP)
#LET pp=$(pp)+100
#GOTO Alfa, $(pp)<500
```

Facciamo rilevare la possibilità, anche all'interno delle regole di creare un'etichetta contenete delle istruzioni.

4.6.20 Script *#VB e #ENDVB*

Questi due comandi permettono di scrivere delle righe di codice con un linguaggio di Script all'interno della regola. Questo tipo di applicazione non trova molto riscontro in ambito grafico, ma piuttosto per elaborazioni particolari legate a gestioni anagrafiche o alle stampe.

Grazie a questi comandi si può accedere a tutte le funzioni previste dal ***VBSCRIPT*** (al momento è l'unico linguaggio di script da noi collaudato).

La proprietà V() permette la comunicazione tra i due ambienti (regole e script) attraverso il passaggio di variabili di input e output.

Esempio 25. Inserimento degli script all'interno di una regola.

Codice 103. Comando *#VB* (1).

```
#VB
  V("R1")= EX(V("str"),",",1) 'corrisponde a $(str:1)
  V("R2")= EX(V("str"),",",2) 'corrisponde a $(str:1)
  V("R3")= EX(V("str"),",",3) 'corrisponde a $(str:2)
  V("R4")= EX(V("str"),",",4) 'corrisponde a $(str:3)
#ENDVB

#MSG $(r1)!$(r2)!$(r3)!$(r4)
```

In questo esempio, per estrarre i vari campi di una variante (STR) è stata usata la proprietà V() nel modo:

EX(V(nomevariante),,,nrcampo)

mentre per usare una variabile normale:

V(nomevariabile) .

Codice 104. Comando *#VB* (2).

```
#LETS XVAR=$(90:6)
#VB
   V("R1")= EX( EX(V("XVAR"),";",2) ,"=",2)
   V("R2")= EX( EX(V("XVAR"),";",3) ,"=",2)
   V("R3")= EX( EX(V("XVAR"),";",4) ,"=",2)
```

```
#ENDVB
#IF $(R1)==$(R2) & $(R1)==$(R3)
    #LET INIZIOY=$(INIZIOY)+3
    #LETS DES=$[SQL;SELECT VARIANTI.MEMO FROM VARIANTI WHERE
       VARIANTI.CODOPZ='$(R1)']

    #VB
      V("TESSUTO")= EX(V("DES"),CHR(9),2)
    #ENDVB
    TX TL 10 (18,$(INIZIOY)) TESSUTO $(R1) $(TESSUTO)
    #LET INC=6
#ENDIF
```

Come si può notare la variabile XVAR viene usata all'interno dello script VB. Le variabili create sono R1, R2 ed R3. Possiamo evidenziare nello script anche il doppio utilizzo del comando EX per estrarre dalla variabile XVAR determinati campi interni. Questa funzione (EX) è stata opportunamente implementata nel programma ***3CadEvolution*** per facilitare alcune operazioni usate frequentemente.

La gestione degli script può essere implementata in qualsiasi gruppo di valutazione di una Regola e anche nelle regole del tipo SQR nel menu Elaborazioni.

4.6.21 Comandi *#FORDBS* e *#ENDDBS*

Questo comando consente di scorrere il risultato di una *Query* che ritorna un elenco di dati all'interno di una regola.

Non vanno utilizzati comunque questi comandi quando si vuole rendere la regola compatibile per la Versione internet del programma.

Esempio 26. Utilizzo comando *#FORDBS*.

Codice 105. Scorrere il risultato di una query.

```
#FORDBS alfa=select codopz, des from varianti where codvar='
   colore'
  #MSG $(alfa)
#ENDDBS
```

4.6.22 Comando *#FORFILE*

Questo commando permette di scorrere tutto il contenuto di un file testuale. Il file da leggere deve essere localizzato nella cartella del catalogo, possibilmente con l'estensione .TXT .

Il comando *#FORFILE* funziona allo stesso modo del comando *#FOR*.

Esempio 27. Utilizzo comando *#FORFILE*.

Codice 106. Scorrere il contenuto di un file testuale.

```
#FORFILE FILE.TXT,,variabile
#MSG $(variabile:0) -- $(variabile:1)
...
#ENDFILE
```

4.7 Comandi estesi per la grafica

Qui vediamo l'utilizzo delle regole in ambiente grafico. Questi comandi sono caratterizzati dal carattere ! all'inizio anziché # e non sono mai costrutti logici ma solo funzioni autodefinite all'interno del programma (non vanno su più righe o appaiati etc.)

L'utilizzo di questi comandi permette di estendere le funzionalità delle regole grafiche.

4.7.1 Comandi *!QUOTAxxx* e *!QUOTE*

Questi comandi permettono di quotare gli elementi inseriti in composizione: il primo consente di quotare il singolo articolo, il secondo evidenzia la quota totale. Nella regola, dopo il comando bisogna evidenziare il punto di inizio e fine quota, l'eventuale misura se diversa da $L e la distanza dal disegno della quota stessa.

La sintassi da usare è la seguente:

!QUOTA[N] (x1 y1 z1), (x2 y2 z2), [testo], LIV

Omettendo il parametro [testo] viene visualizzata la distanza effettiva tra i due punti. Mentre il parametro LIV indica la distanza del disegno dai punti quotati.

Esempio 28. Comando *!QUOTA*.

Codice 107. Inserimento di una quota.

```
!QUOTA (0 $A 0), ($L $A 0), ,100
```

La dimensione che si andrà a quotare inizia dall'origine sinistra in alto del mobile e termina in corrispondenza del lato destro. Il disegno della quota sarà spostato di 10 cm rispetto alla composizione. *!QUOTAJ* usa gli stessi parametri e sintassi.

Se si vogliono differenziare vari tipi di quotatura [N], ad esempio quella delle basi rispetto a quella dei pensili, basta specificare un numero progressivo che identifichi il tipo di quota.

Esempio 29. Due quote differenti.

Codice 108. Differenziare vari tipi di quotatura.

```
1 !QUOTA1 (0 $A 0),($L $A 0), ,100
2 !QUOTA2 (0 $A 0),($L $A 0), ,150
```

Nella fase di stampa sarà poi possibile decidere quali quote far comparire. Se una o più coordinate da quotare sono il risultato di un'operazione algebrica, questa va racchiusa tra parentesi graffe. Esempio:

Codice 109. Operazioni algebriche nelle quotature.

```
1 !QUOTA (0 $A 0),({$L/2} $A 0),,100
2 !QUOTAP (x1 y1 z1), (x2 y2 z2), [testo], LIV
```

Permette alla quota di interagire tra un elemento caricato in grafica e la parete caricata. Indicherà la distanza tra l'oggetto e la parete.

La sintassi da usare è la seguente:

!QUOTAT (x1 y1 z1), (x2 y2 z2), [testo], LIV

Restituisce invece la quota totale.

Altro modo per eseguire le quotature: creare una tipologia apposita, e predisporre nel campo commento della tipologia le dimensioni della quota e le varie informazioni che la riguardano insieme al parametro 1 che indica il tipo di quota.

Esempio 30. Creazione delle quotature senza il comando *!QUOTA*.

Codice 110. Creare le quotature senza il comando *!QUOTA*.

```
1 800, lato=9, COD= //$L, SL=0, SA=-100, SP=$P, L=$L, A=10, P
    =5, X1=2
```

Modificando il parametro X1, sarà possibile visualizzare diversi tipi di quote. Sta al programmatore scegliere la più opportuna alla propria esigenza. Qui di seguito l'elenco delle possibili variazioni di quote:

1= Testo riquadrato inserito al centro

2= Quota

3= Testo riquadrato inserito a sinistra

4= Testo riquadrato in alto al centro del monitor grafico

5= Testo riquadrato tipo fumetto in alto al centro del monitor grafico

6= Testo riquadrato in alto a sinistra del monitor grafico

7= Testo riquadrato tipo fumetto in alto a sinistra del monitor grafico

8= Quota spezzata tipo direttrice

Dalla versione 600 vengono introdotti una serie di miglioramenti funzionali ai comandi di quotatura *!QUOTAxx*, pur mantenendo la totale compatibilità con le quote costruite precedentemente.

Questi miglioramenti cercano di superare alcune precedenti limitazioni del sistema:

1. I comandi QUOTA dovevano indicare punti con posizionamento assoluto nel BOX, anche se si stava quotando all'interno di una macro o di un divider che hanno spostamenti di origine, con l'ovvia implicazione di dover rifare a mano i conti.
2. I punti venivano indicati nella forma (x y z), usando come separatore lo spazio, implicando problemi quando si utilizzavano variabili o formule.

Per la prima limitazione è stato introdotto il comando *!QUOTE*.

!QUOTE

Questo comando va messo all'inizio del sistema di quotatura, dovrebbe risiedere nella macro ASSEGNA e impostato solo una volta.

La sintassi da usare è la seguente:

!QUOTE modo, ox, oy, oz

Se viene impostato modo=1 il posizionamento delle quote, come con le tipologie, avviene tramite uno stack di matrici di trasformazione (quindi le quote sono relative alla macro in cui ci si trova); E' possibile inoltre definire una origine alternativa in ox, oy, oz.

Esempio 31. Utilizzo comando *!QUOTE*.

```
1 !QUOTE 1, $nl, $na, $np
```

Per la seconda limitazione si è previsto di indicare sintassi alternative per i punti. Si confrontino questi esempi tra di loro equivalenti nella nuova versione.

Esempio 32. Quota con la sintassi alternativa per i punti.

```
!QUOTA1 (10 0 0),($l 0 0),$l-10,10
!QUOTA1 (10,0,0),($l,0,0),$l-10,10
!QUOTA1 (10;0;0),($l;0;0),$l-10,10
!QUOTA1 (10;0;0),X$l,,10
!QUOTA1 (10;0;0),X+$l-10,,10
```

Per il secondo punto di quotatura può essere utilizzata oltre alla sintassi (<x>,<y>,<z>) anche *X<valorex>*, che indica che il secondo punto è uguale al primo tranne che per la componente X (equivalente *Y<valore>* e *Z<valore>*).

Scrivendo *X+<valore>* si ottiene che il secondo punto è spostato in x rispetto al primo di <valore>.

Sono disponibili anche comandi *!QUOTAL* , *!QUOTAC* e *!QUOTAR*.

La sintassi da usare è la seguente:

!QUOTA<L,C,R> testo, x1, y1, x2, y2, scala, div

L : il testo specificato viene posizionato a sinistra della quota;

C : il testo specificato viene posizionato al centro della quota;

R : il testo specificato viene posizionato a destra della quota.

Inoltre è possibile specificare la scala del disegno.

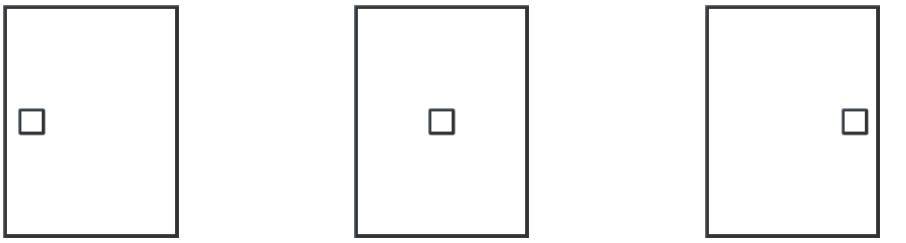
4.7.2 Comando *!PM*

Questo comando consente di forzare il verso sinistro o destro in una regola. Nel settore del mobile il verso è determinato dall'apertura dell'anta, se esiste, oppure se ad esempio si tratta di un elemento terminale il verso è dato dal suo posizionamento nella composizione.

La Figura 4.6 nella pagina 133 mostra rispettivamente un'anta destra, una centrale e una sinistra.

Codice 111. Forzare il verso.

```
!PM 1 // per forzare il verso destro
!PM 2 // per forzare il verso centrale
!PM 3 // per forzare il verso sinistro
```



(1) Verso destro (2) Verso centrale (3) Verso sinistro

PM=1 PM=2 PM=3

Figura 4.6: Posizione anta.

4.7.3 Comando *!ALTEZZE*

Questo comando è legato alla gestione delle linee altezze nei flag speciali (v. paragrafo relativo), in particolare si usa quando vi sono più modulistiche definite. E' necessario specificare il numero della colonna a cui fare riferimento, tale numero può essere sostituito da una variabile.

Codice 112. Gestire delle linee altezze.

```
1 !ALTEZZE 1
```

4.7.4 Comandi *!MTTRASLA* e *!MTINIT* o *!MTRESET*

Con *!MTTRASLA* possiamo appunto traslare una o più tipologie nelle tre direzioni e ruotarle nei tre assi, con *!MTINIT* o *!MTRESET* (sono equivalenti) si annullano gli spostamenti precedentemente definiti.

La sintassi da usare è la seguente:

!MTTRASLA L,A,P,AX,AY,AZ !MTINIT oppure !MTRESET

Codice 113. Traslare una o più tipologie e poi annullare lo spostamento.

```
1 !MTTRASLA 0,0,0,0,45,0
2 5 // anta
3 2 // base
4 4 // ripiano
5 !MTRESET
```

Nell'esempio abbiamo impostato una rotazione sull'asse y di 45 gradi, tutte le tipologie che inseriamo dopo assumeranno tali impostazioni, fino a quando azzeriamo gli spostamenti con il comando *!MTRESET* (o *!MTINIT*).

4.7.5 Comandi *!ADDPUNTO*, *!DELPUNTO* e *!AZZERAPUNTI*

In relazione alla quotatura manuale degli elementi (v. manuale utente), sono stati introdotti questi comandi che consentono di aggiungere e togliere punti significativi da quotare.

La sintassi prevede di specificare le coordinate x, y, z del punto da inserire o eliminare. Come già precisato in precedenza, l'eventuale espressione algebrica che rappresenta una coordinata va racchiusa tra parentesi graffe.

Codice 114. Aggiungere o togliere punti significativi da quotare.

```
1 !ADDPUNTO X,Y,Z
2 !DELPUNTO X,Y,Z
3 !AZZERAPUNTI
```

senza alcun parametro elimina tutti i punti, sia quelli standard che quelli inseriti con *!ADDPUNTO*.

4.7.6 Comandi *!INVERITLATO* e *!AZZERAINVERTILATO*

Questi due comandi permettono di capovolgere tutte le considerazioni fatte all'interno di una regola sul $PM (senso sinistro o destro). Per chiarire il concetto facciamo riferimento alla base angolare delle cucine (regola per la quale sono stati creati i due comandi).

Data la complessità di gestione dell'angolo fuori squadra, la regola è stata costruita dagli operatori della Dau S.r.l.. Siccome non c'è omogeneità tra i produttori di mobili nella definizione della base angolare destra e sinistra (per alcuni la base sinistra è quella con l'anta con apertura destra), per evitare di generare un'altra regola con gli opportuni controlli sul senso ($PM) invertiti, è stato introdotto il comando *!INVERITLATO*, che va inserito nella regola originale nel caso in cui il senso sia contrario a quello da noi considerato. E' importante che alla fine della regola si inserisca anche *!AZZERAINVERTILATO* per riportare la valutazione del $PM alla normale gestione.

4.7.7 Comando *!COMMNETO*

Questo comando consente di inserire nei codici articoli richiamanti dei commenti o note.

La sintassi da usare è la seguente:

!COMMENTO nota1, nota2, nota3

Esempio 33. Utilizzo comando *!COMMNETO*.
Inserisce il commento sulla terza riga di annotazione.

Codice 115. Inserire una nota su una riga specifica.

```
1 !COMMENTO3 sulla terza riga
```

4.7.8 Comando !FLAGSPEC

Questo comando consente di impostare il quarto carattere del flag speciale legato all'articolo a un valore particolare per dichiarare che l'articolo è fuorimisura.

Codice 116. Dichiarare un articolo come fuori misura.

```
1 !FLAGSPEC T
```

4.7.9 Comandi per le informazioni sull'articolo

Vi sono una serie di comandi per richiedere informazioni circa il codice di articolo che ha richiamato la regola:

Codice 117. Informazioni sull'articolo.

```
1 !CODICE var00 // inserisce in var00 il codice articolo
2 !MODELLO var01 // inserisce in var01 il modello
3 !NEUTRO var02 // inserisce in var02 il codice neutro
4 !VARIANTE var03 STR // inserisce in var03 il valore gest.
    della variante STR
```

4.7.10 Comandi *!ADDLINK*, *!SCEGLIADDLINK*, *!TIPOBOX*

Questa funzione consente di creare diversi collegamenti (sempre tra box) a seconda delle necessità (quindi collegamenti diversi con box diversi) sfruttando una semplice sintassi e seguendo sempre la stessa tabella sopra riportata.

Il metodo di lavoro da utilizzare consiste nel denominare ogni tipo di box con il comando *!TIPOBOX* e un numero identificativo del box stesso. Dopodiché si definisce con la funzione *!ADDLINK* il vero e proprio tipo di collegamento che si vuole instaurare.

Codice 118. Creare diversi collegamenti tra i box.

```
1 !AZZERA
2 !TIPOBOX 1
3 !ADDLINK1 coll a dx,101
4 !ADDLINK2 coll sopra,501
```

Con questo metodo abbiamo definito due diversi tipi di link per lo stesso box ovvero: nel caso in cui si affianchino due box sopralzo ci sarà fra loro un collegamento a dx (101 come da tabella), quando si aggiungerà una schiena questa verrà posta sopra (501 come da tabella).

Al comando *!ADDLINK* possono essere aggiunti dei parametri facoltativi che specificano degli spostamenti nelle tre direzioni L, A, P ed una eventuale rotazione.

La sintassi da usare è la seguente:

!ADDLINK5 coll sopra, 501, xx, yy, zz, aa

dove:

- Una breve descrizione chiarisce il tipo di collegamento;
- 501 è il codice che individua il tipo di collegamento (tabellato vedi sotto);
- xx è l'eventuale spostamento in larghezza;
- yy è l'eventuale spostamento in altezza ;
- zz è l'eventuale spostamento in profondità ;
- aa è l'eventuale rotazione.

Link

Numero Link	Descrizione collocamento
101	destro base retro
102	destro base fronte
103	destro ponte retro
104	destro ponte fronte
201	sinistro base retro
202	sinistro base fronte
203	sinistro ponte retro
204	sinistro ponte fronte
301	fronte base sinistra
302	fronte base destra
303	fronte ponte sinistra
304	fronte ponte destra
401	retro base sinistra

Tabella 4.6 : continua nella prossima pagina

Tabella 4.6 : continua dalla pagina precedente

Numero Link	Descrizione collocamento
402	retro base destra
403	retro ponte sinistra
404	retro ponte destra
501	sopra retro sinistra
502	sopra retro destra
503	sopra fronte sinistra
504	sopra fronte destra
601	sotto retro sinistra
602	sotto retro destra
603	sotto fronte sinistra
604	sotto fronte destra

Tabella 4.6: Link utilizzati dal comando *!ADDLINK*.

Per questo primo gruppo di link si nota una certa convenzione nella generazione dei numeri di link. La prima cifra del numero corrisponde ai seguenti lati di collocamento:

1= destro

2= sinistro

3= fronte

4= retro

5= sopra

6= sotto

La seconda cifra 0 indica che non ci sono variazioni angolari per il posizionamento dell'elemento.

L'ultima cifra numera i possibili collocamenti all'interno del gruppo corrispondente alla prima cifra.

Link Gruppo

Numero Link	Descrizione collocamento
111	90 gradi sullo spigolo destro
112	90 gradi fronte destro

Tabella 4.7 : continua nella prossima pagina

Tabella 4.7 : continua dalla pagina precedente

Numero Link	Descrizione collocamento
113	90 gradi lato destro
211	90 gradi sullo spigolo sinistro
212	90 gradi fronte sinistro
213	90 gradi lato sinistro
121	-90 gradi sullo spigolo destro
122	-90 gradi dietro destra
123	-90 gradi lato destro
221	-90 gradi sullo spigolo sinistro
222	-90 gradi dietro sinistra
223	-90 gradi lato sinistro
411	180 gradi contrapposto
131	60 gradi sullo spigolo destro
132	60 gradi fronte destro
133	60 gradi lato destro
231	60 gradi sullo spigolo sinistro
232	60 gradi fronte sinistro
233	60 gradi lato sinistro
141	45 gradi sullo spigolo destro
142	45 gradi fronte destro
143	45 gradi lato destro
241	45 gradi sullo spigolo sinistro
242	45 gradi fronte sinistro
243	45 gradi lato sinistro
151	30 gradi sullo spigolo destro
152	30 gradi fronte destro
153	30 gradi lato destro
251	30 gradi sullo spigolo sinistro
252	30 gradi fronte sinistro
253	30 gradi lato sinistro
111	90 gradi sullo spigolo destro
112	90 gradi fronte destro

Tabella 4.7 : continua nella prossima pagina

Tabella 4.7 : continua dalla pagina precedente

Numero Link	Descrizione collocamento
113	90 gradi lato destro
211	90 gradi sullo spigolo sinistro
212	90 gradi fronte sinistro
213	90 gradi lato sinistro
121	-90 gradi sullo spigolo destro
122	-90 gradi dietro destra
123	-90 gradi lato destro
221	-90 gradi sullo spigolo sinistro
222	-90 gradi dietro sinistra
223	-90 gradi lato sinistro
411	180 gradi contrapposto

Tabella 4.7: Numeri per i collocamenti all'interno del gruppo.

Anche in questo secondo gruppo di link si può notare una certa convenzione nella generazione dei numeri di link. La prima cifra del numero corrisponde ai seguenti lati di collocamento:

1 =destro

2 =sinistro

3 =fronte

4 =retro

5 =sopra

6 =sotto

La seconda cifra indica l'angolo di collocamento dell'elemento:

1 =90 gradi

2 =-90 gradi

3 =60 gradi

4 =45 gradi

5 =30 gradi

L'ultima cifra numera i possibili collocamenti all'interno del gruppo corrispondente alle prime due cifre.

Esempio 34. Posizionamento box a 90 gradi sullo spigolo destro.

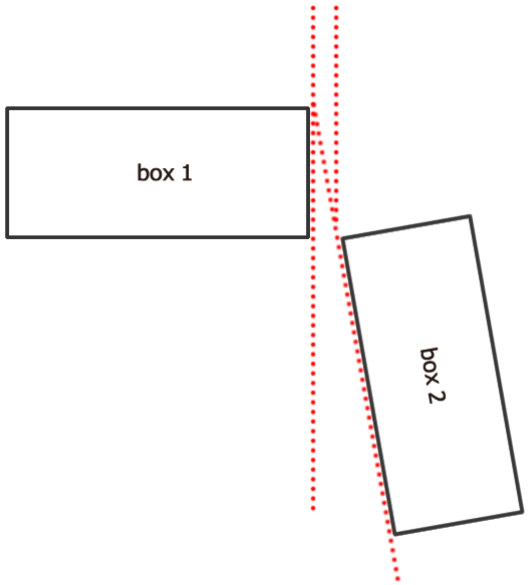


Figura 4.7: Rotazione box.

Supponendo di essere in visualizzazione pianta vogliamo posizionare il nuovo box (2) usando il Tipo Link 111 (vedi tabella 4.6) ma con uno spostamento in larghezza di 20 mm e uno spostamento angolare di 10° rispetto al box precedente.

4.7.11 Comando *!ADDMAGIC*

Questo comando crea un Magic Link per i box di tipo <N> che permette di collegare box simili su linee base che vanno da <p1> a <p2> con l'avvicinamento dell'oggetto.

La sintassi da usare è la seguente:

!ADDMAGIC<N> <p1x>, <p1z>, <p2x>, <p2z>, <alt>

Esempio 35. Utilizzo comando *!ADDMAGIC*

Codice 119. Collegare box simili su linee base.

```
!ADDMAGIC1 0, 0, 0, $P
!ADDMAGIC1 $l, 0, $l, $P
```

Nell'uso con drag&drop degli articoli vengono visualizzate due linee tratteggiate che indicano i due punti di aggancio disponibili per i due box in avvicinamento.

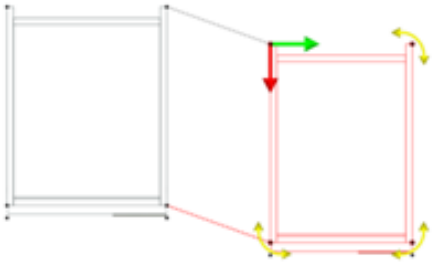


Figura 4.8: Collegamento box su linee base 1.

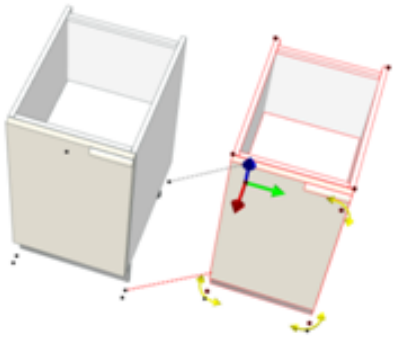


Figura 4.9: Collegamento box su linee base 2.

4.7.12 Comando *!IMGSCALA*

Questo comando permette di leggere un'immagine e impostare il rapporto tra dimensione x e y in una variabile.

Esempio 36. Utilizzo comando *!IMGSCALA*.

Codice 120. Leggere un'immagine.

```
1 !IMGSCALA TEST, $(.percorso)\foto\immagine.jpg
2 #LET A=$L/$(TEST)
```

4.7.13 Comando *!LASER*

Il comando *!LASER* permette di controllare quanto un oggetto è vicino a un altro, proiettando un raggio dal primo al secondo.

Questo riporta la distanza in mm dall'oggetto trovato o il valore 999999 se non ha trovato oggetti.

Tale comando è stato integrato con un parametro che specifica l'elenco delle tipologie da controllare.

La sintassi da usare è la seguente:

!LASER [var], [x1], [y1], [z1], [x2], [y2], [z2], [elencotipologie]

Questa funzione inserisce nella variabile [variabile], la distanza minima di un oggetto appartenente a una tipologia definita nell'elenco (separatore , o ;) per un raggio che parte dal punto x1, y1, z1 e passa per il punto x2, y2, z2 .

! Se non si specifica l'elenco delle tipologie, questa funzione cerca solo gli oggetti di tipo collisione (lato 23), altrimenti il controllo è fatto sulle tipologie indicate.

4.7.14 Comandi *!CODICEBARRA*, *!DIMBARRA*

Tali comandi permettono di ricavare proceduralmente le informazioni sul codicebarra associato a una determinata misura e informazioni sulle dimensioni standard più piccole e più grandi all'interno del codice barra.

!CODICEBARRA[6]

La sintassi da usare è la seguente:

!CODICEBARRA [nomevariabile], [codicebarra], [l], [a], [p]

questo comando carica nella variabile [nomevariabile] il codice articolo generato a partire da un codice barra e dalle dimensioni specificate. La variabile è strutturata in questo modo:

- $(nomevariabile:0) : è il codice dell'articolo;
- $(nomevariabile:1) : è la dimensione L anagrafica dell'articolo;
- $(nomevariabile:2) : è la dimensione A anagrafica dell'articolo.

!DIMBARRA

La sintassi da usare è la seguente:

!DIMBARRA [nomevariabile],[codicebarra],[misura],[tipo]

questo comando carica nella variabile [nomevariabile] in funzione del tipo di misura [tipo]:

L+ : dimensione larghezza superiore standard
L- : dimensione larghezza inferiore standard
A+ : dimensione Altezza superiore standard
A- : dimensione altezza inferiore standard

Esempio 37. Utilizzo comandi *!CODICEBARRA*, *!DIMBARRA*
.

```
1 !BARRA ALFA,^XRS3002,1090,400,20
2 !DIMBARRA A01,^XRS3002,1090,L+
3 !DIMBARRA A02,^XRS3002,1090,L-
4 !DIMBARRA A03,^XRS3002,400,A
5 !DIMBARRA A04,^XRS3002,400,A-
6 #MSG $(alfa:0)|$(alfa:1)|$(alfa:2)|$(A01)|$(A02)|$(A03)|$(A04
    )
```

4.8 Comandi per la gestione delle sagome

4.8.1 Comando *!SAGOMA*

La forma tipica di un tipologia è un rettangolo. Per creare un oggetto di forme diverse è necessario utilizzare il comando *!SAGOMA* all'interno della regola. Nel procedimento per prima cosa si definisce la forma (sagoma) che si vuole dare alla tipologia e poi questa viene associata alla tipologia.

Una sagoma rappresenta un oggetto bidimensionale, chiuso che viene costruito attraverso dei punti. Il comando *!SAGOMA* traccia delle linee che congiungono i punti passati come parametri.

La sintassi da usare è la seguente:

!SAGOMA sagoma, x1, y1, x2, y2, ...

Codice 121. Definizione di una sagoma semplice.

```
1 !SAGOMA sagoma 0,0 , $L, $P, 0, $P
```

Essendo sagoma un oggetto chiuso se il punto iniziale ed il punto finale della sagoma non coincidono viene tracciata una linea.

Associazione della sagoma alla tipologia: Per visualizzare un oggetto sagomato nella grafica è necessario prima definire una sagoma e poi associarla ad una tipologia, ciò avviene attraverso l'assegnazione della sagoma alla variabile *DIS*.

Codice 122. Associazione della sagoma al disegno.

```
1 SAGOMA sagoma 0,0 , $L, $P, 0, $P
2 001,DIS=$(sagoma)
```

Inoltre è possibile associare la sagoma tramite l'interfaccia della gestione della tipologia.

Dall'editor della regola sulla riga della tipologia cliccare tasto F3 e nel campo Disegno , Figura 4.10 nella pagina 144, inserire il nome della sagoma. In questo modo però l'associazione è statica e ogni volta che viene richiamata la tipologia 001, viene visualizzata la sagoma.

Questo è comodo quando c'è la necessità di utilizzare una sagoma spesso perciò si crea una tipologia e si associa una sagoma statica.

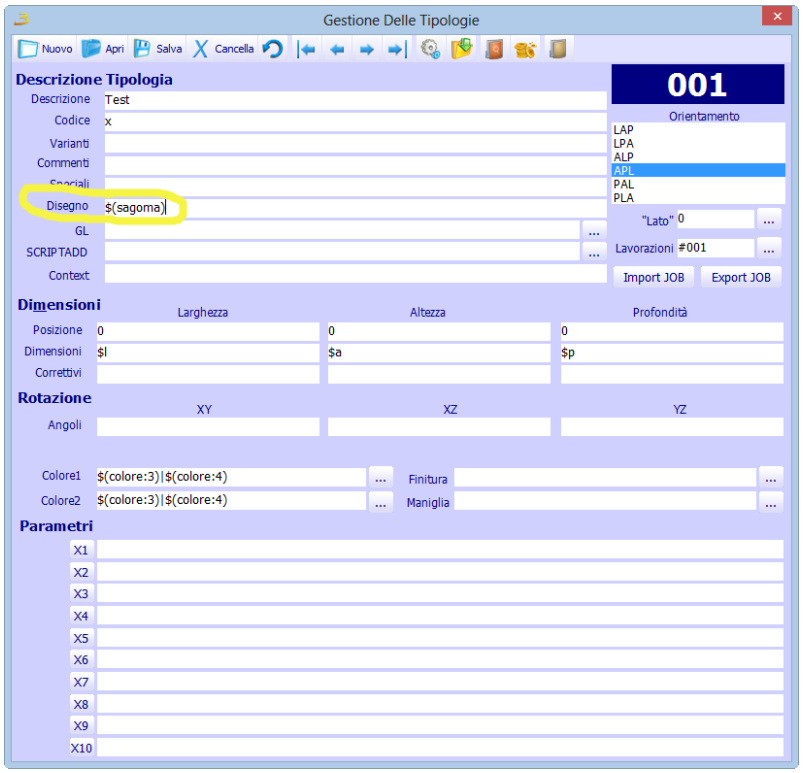


Figura 4.10: Editor tipologia, associazione della sagoma.

Precedentemente abbiamo creato una sagoma che passa per tre punti e l'abbiamo associata alla tipologia 001. Come possiamo vedere dalla figura Figura 4.11 nella pagina 144 la sagoma che abbiamo creato ha la forma a spigoli, perché in questa sagoma i punti che abbiamo definito vengono tracciati attraverso le linee rette. E' possibile definire una sagoma con degli archi tramite inserimento del comando *A[Numero segmenti]* nella definizione della sagoma.

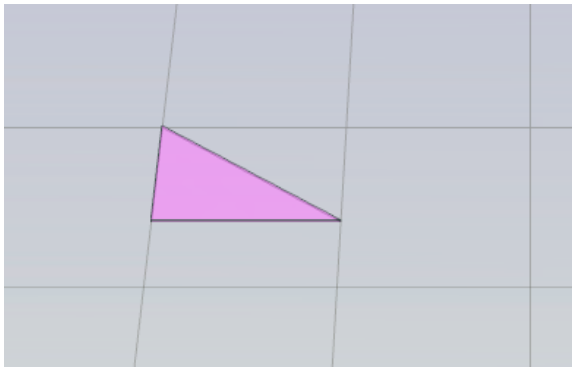


Figura 4.11: Sagoma semplice.

Esempio 38. Sagoma con arco(1).

Codice 123. Utilizzo comando A nella costruzione della sagoma(1).

```
1 !SAGOMA sagomaConArchi 0,0, A30, 0,0, $l/2,$P/3,$L,$P,0,$P
```

Il comando che definisce l'arco, oltre al numero dei segmenti che lo compongono, necessita di tre punti : il primo è il punto di partenza nel nostro esempio è 0,0; il secondo punto è il punto centrale per il quale deve passare l'arco, nel nostro esempio è $l/2, $P/3 mentre il terzo punto è il punto di fine arco, nel nostro esempio è ,$L,$P.

Esempio 39. Sagoma con arco(2).

Codice 124. Utilizzo comando A nella costruzione della sagoma(2).

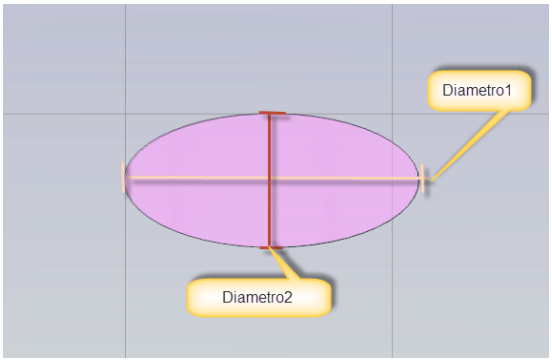
```
1 !SAGOMA sagomaConArchi A30, 0,0, $l/2,-$P,$L,$P,0,$P
```

Abbiamo visto che una sagoma può essere rappresentata da linee ed archi. Il comando *!SAGOMA* accetta però anche il comando *E* per disegnare una sagoma a forma di ellisse o il comando *C* per disegnare una sagoma a forma di cerchio.

Esempio 40. Sagoma a forma di ellisse.

Codice 125. Utilizzo comando E nella costruzione della sagoma.

```
1 !SAGOMA sagomaFormaEllisse E50, 0,0, 550, 250
```



Il comando E[numero segmenti] oltre al numero di segmenti accetta anche il punto d'origine dell'ellisse e le due misure *Diametro1* e *Diametro2* per designare l'ellisse.

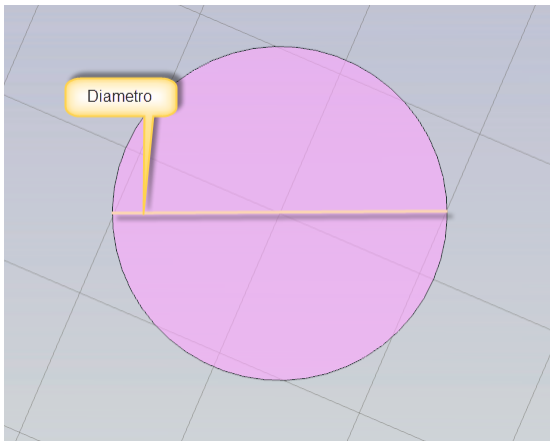
Figura 4.12: Sagoma a forma di ellisse.

Esempio 41. Sagoma a forma di cerchio.

Codice 126. Utilizzo comando C nella costruzione della sagoma.

```
1 !SAGOMA sagomaFormaCerchio C50, 0,0, 550
```

Inoltre è possibile applicare delle estrusioni agli oggetti sagomati.



Il comando C[numero segmenti] oltre al numero di segmenti accetta anche il punto d'origine del cerchio ed il diametro per designare il cerchio.

Figura 4.13: Sagoma a forma di ellisse.

In base al tipo di estrusione che si vuole ottenere si possono usare diversi comandi di tipo Sagoma:

!SagomaP: estrusione secondo la profondità;

!SagomaL: estrusione secondo larghezza;

!SagomaC: doppia estrusione (Cappa) secondo altezza;

!SagomaD: doppia estrusione secondo profondità;

!SagomaE: doppia estrusione secondo altezza.

Esempio 42. Utilizzo comando *!SagomaC*.

Codice 127. Estrusione.

```
#LET l=1000,p=500
!SAGOMAC sagoma 0,0,1000,0,1000,500,0,500, \
       100,0,300,0,300,100,100,100

001,DIS=$(sagoma)
```

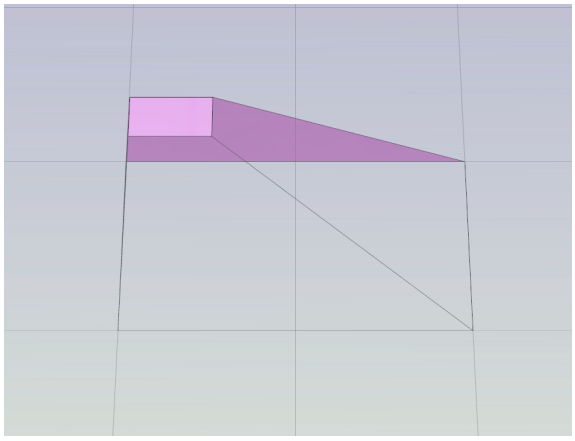


Figura 4.14: Doppia estrusione della sagoma secondo l'altezza.

4.8.2 Comando *!SAGDUMP*

Se il comando *!SAGOMA* definisce una sagoma a partire da un insieme di punti, il comando *!SAGDUMP* fa esattamente il contrario e cioè data una sagoma definisce l'insieme dei punti di costruzione della stessa. Questo comando è utile per riprodurre parti di sagome complesse. La figura Figura 4.15 nella pagina 147 riporta output del codice 128.

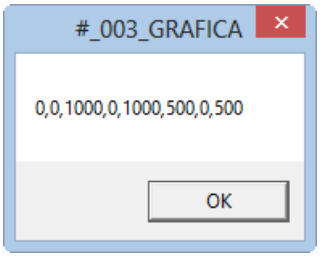


Figura 4.15: Output prodotto dal comando !SAGDUMP.

Codice 128. Comando !SAGDUMP.

```
1 !SAGOMA sagoma 0,0,1000,0,1000,500,0,500
2 !SAGDUMP VAR,$(sagoma)
3 #MSG $(VAR)
```

4.8.3 Comandi *!SAGMID, !SAGPERCX, !SAGPERCY*

Il comando *!SAGMID*, partendo da una sagoma e due coefficienti, restituisce una nuova sagoma compresa tra i due valori.

La sintassi da usare è la seguente:

!SAGMID nuovaSagoma, coefficiente1, coefficiente2, sagoma

I due parametri devono essere compresi tra 0 e 1 e rappresentano le percentuali rispetto alla lunghezza della sagoma dove 0 rappresenta il punto iniziale della sagoma e l'1 punto finale, cioè la sagoma restituita tra i coefficienti 0 e 1 è la sagoma stessa.

Siccome il comando *!SAGMID* accetta le percentuali rispetto alla lunghezza della sagoma è necessario utilizzare i comandi *!SAGPERCX* e *!SAGPERCY* per calcolare a che percentuale si trova un punto rispetto l'asse x o rispetto l'asse y. Dalla figura Figura 4.16 nella pagina 148 si può vedere a che percentuale si trova il punto che passa per la metà della larghezza della tipologia, codice 129.

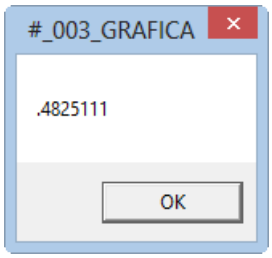


Figura 4.16: Output comando !SAGPERCX.

Codice 129. Utilizzo comando !SAGPERCX.

```
1 !SAGOMA TMP 0,0, A30, 0,0, $l/2,$P/3,$L,
      $P,0,$P
2 !SAGPERCX percRispettoX, $l/2,$(TMP)
3 #MSG $(percRispettoX)
```

La sintassi da usare è la seguente:

!SAGPERCX percentuale, posX, sagoma

La sintassi da usare è la seguente:

!SAGPERCY percentuale, posY, sagoma

4.8.4 Comando *!SAGXY*

Questo comando restituisce nelle variabili varx e vary il valore x ed il valore y del punto che si trova alla percentuale rispetto la lunghezza della sagoma specificata dalla variabile coefficiente. Il valore del coefficiente deve essere compreso tra 0 ed 1.

La sintassi da usare è la seguente:

!SAGXY varx, vary, coefficiente, sagoma

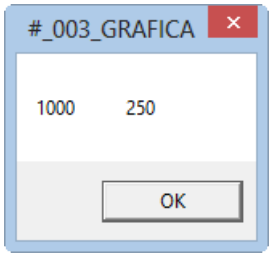


Figura 4.17: Output comando !SAGXY.

Codice 130. Comando !SAGXY.

```
1 !SAGOMA sagoma 0,0,1000,0,1000,500,0,500
2 !SAGXY VARX, VARY, 0.5, $(sagoma)
3 #MSG $(VARX), $(VARY)
```

4.8.5 Comando *!SAGOFFSET*

Il comando *!SAGOFFSET* espande la sagoma attorno al suo percorso di un profilo.

L'offset può assumere anche valori negativi, producendo un questo modo una contrazione della sagoma, anche se il risultato potrebbe non essere sempre corretto in questo caso. Ingrandisce il profilo della sagoma di un particolare valore di offset.

La sintassi da usare è la seguente:

!SAGOFFSET sagomaNuova sagoma valoreOffset

Esempio 43. Utilizzo comando *!SAGOFFSET*.

Codice 131. Ingrandire il profilo della sagoma.

```
1 !SAGOFFSET TMP1 $(tmp) 30
```

Fa un offset di 30 mm.

4.8.6 Comandi *!SAGCLIPX, !SAGCLIPY*

Il comando *!SAGCLIPX* taglia la sagoma lungo l'asse X da x1 a x2.

La sintassi da usare è la seguente:

!SAGCLIPX sagomaNuova, sagoma, x1, x2

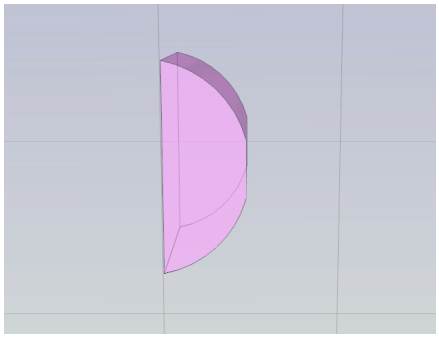


Figura 4.18: Taglio della sagoma lungo l'asse X.

Codice 132. Comando !SAGCLIPX.

```
1 !SAGOMA sagoma1 c40, 0,0, 250
2 !SAGCLIPX sagoma, $(sagoma1), 50,240
3 001,DIS=$(sagoma)
```

Il comando *!SAGCLIPY* taglia la sagoma lungo l'asse Y da y1 a y2.

La sintassi da usare è la seguente:

!SAGCLIPY sagomaNuova, sagoma, y1, y2

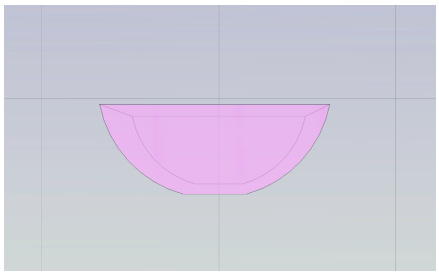


Figura 4.19: Taglio della sagoma lungo l'asse Y.

Codice 133. Comando !SAGCLIPY.

```
1 !SAGOMA sagoma1 c40, 0,0, 250
2 !SAGCLIPY sagoma, $(sagoma1), 50,240
3 001,DIS=$(sagoma)
```

4.8.7 Comandi *!SAGCLIPR, !SAGCLIPL, !SAGCLIPU, !SAGCLIPD*

Il comando *!SAGCLIPR* produce la controsagoma della parte destra della sagoma partendo dal punto x.

La sintassi da usare è la seguente:

!SAGCLIPR sagomaNuova, sagoma, x

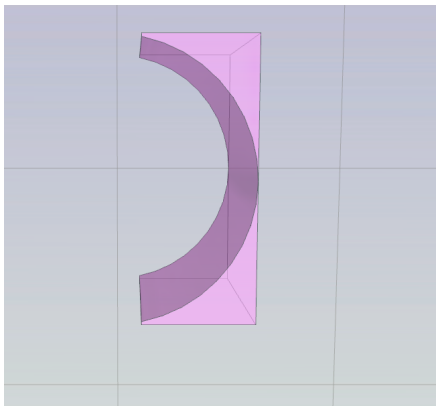


Figura 4.20: Controsagoma della parte destra della sagoma.

Codice 134. Comando !SAGCLIPR.

```
1 !SAGOMA sagoma1 c40, 0,0, 250
2 !SAGCLIPR sagoma, $(sagoma1), 50
3 001,DIS=$(sagoma)
```

Il comando *!SAGCLIPR* produce la controsagoma della parte sinistra della sagoma partendo dal punto x.

La sintassi da usare è la seguente:

!SAGCLIPL sagomaNuova, sagoma, x

Il comando *!SAGCLIPU* produce la controsagoma della parte alta della sagoma partendo dal punto Y.

La sintassi da usare è la seguente:

!SAGCLIPU sagomaNuova, sagoma, y

Il comando *!SAGCLIPD* produce la controsagoma della parte bassa della sagoma partendo dal punto x.

La sintassi da usare è la seguente:

!SAGCLIPD sagomaNuova, sagoma, y

Il comando *!SAGCLIPT* produce la controsagoma dell'intera sagoma distanziata di una misura specificata dall'*offset*.

La sintassi da usare è la seguente:

!SAGCLIPT sagomaNuova, sagoma, offset

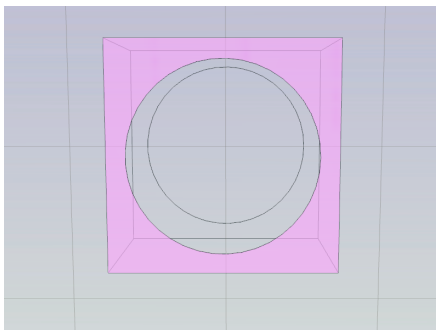


Figura 4.21: Controsagoma dell'intera sagoma distanziata 50mm.

Codice 135. Comando !SAGCLIPT.

```
1 !SAGOMA sagoma1 c40, 0,0, 250
2 !SAGCLIPT sagoma, $(sagoma1), 50
3 001,DIS=$(sagoma)
```

4.8.8 Comando *!FRESA*

Permette di creare delle sagome complesse utilizzando come base un rettangolo e delle sagome precedentemente create. In particolare la sagoma complessa viene creata fresando il rettangolo principale seguendo la forma delle sagome. In questo caso le sagome semplici non sono utilizzate per dare la forma al materiale ma per creare il vuoto.

Un possibile utilizzo delle frese è ad esempio la creazione delle cornici o serramenti.

La sintassi da usare è la seguente:

!FRESA sagomaRisultante, dimX, dimY, sagomaDaIncidere1, sagomaDaIncidere2, ..., sagomaDaIncidereN

I valori *sagomaDaIncidere* sono le sagome che vanno a tagliare il rettangolo. Inoltre per ogni sagoma vengono specificati i parametri di posizionamento, separati dal carattere [;] :

- lato: rappresenta il lato del rettangolo su quale verrà posizionata la sagoma, questo parametro puo assumere i seguenti valori:
  - 0
  - 1
  - 2
  - 3
- puntoX: spostamento della sagoma verso la direzione del lato;
- puntoY: spostamento della sagoma dal lato verso il centro del rettangolo;
- rotazione: spostamento angolare della sagoma, questo valore è opzionale.

Le sagome da incidere vengono posizionate rispetto al lato specificato. I valori specificati dai parametri puntoX e puntoY variano per ogni lato. In Figura 4.22 nella pagina 154 vengono rappresentate le direzioni degli spostamenti per ogni lato.

Vediamo degli esempi sull'utilizzo del comando !FRESA.

Esempio 44. Comando *!FRESA*.

In questo esempio abbiamo creato una sagoma circolare di diametro 50 che va a tagliare il rettangolo sul lato 0 nella posizione 50 rispetto la direzione x, Figura 4.23 nella pagina 155.

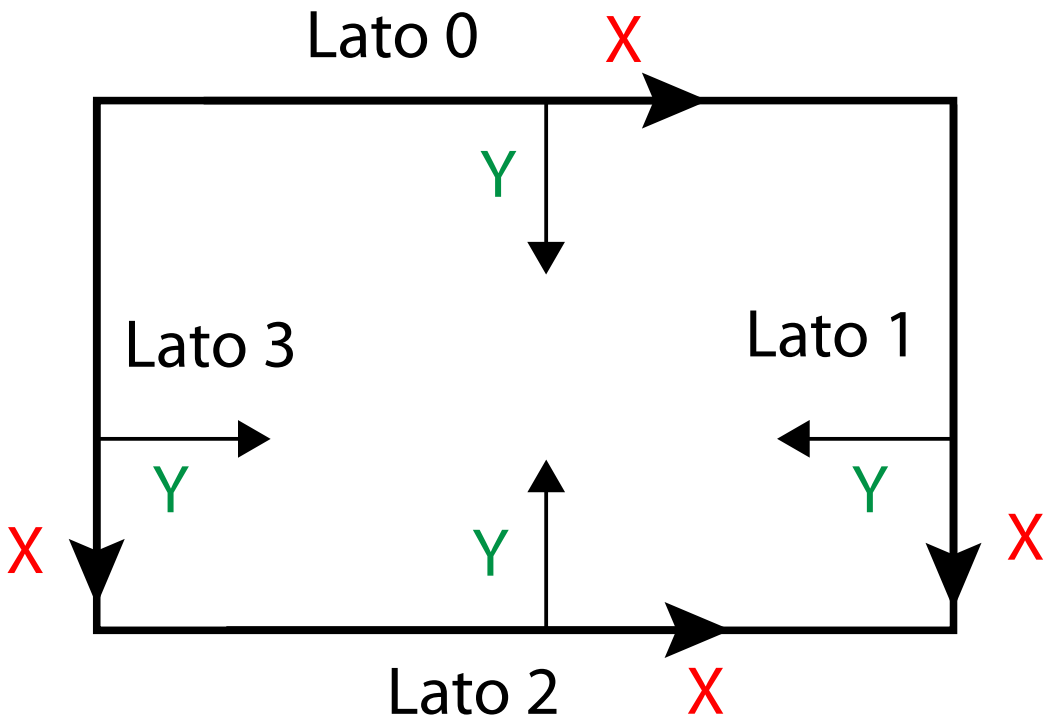


Figura 4.22: Direzioni x e y dei lati per la fresa.

Codice 136. Utilizzo del comando !FRESA.

```
!SAGOMA sag1 C40,0,0,50


!FRESA sagConFresa,500,400, \
      $(sag1);0;50;-10\

1,DIS=$(sagConFresa),A=10
```

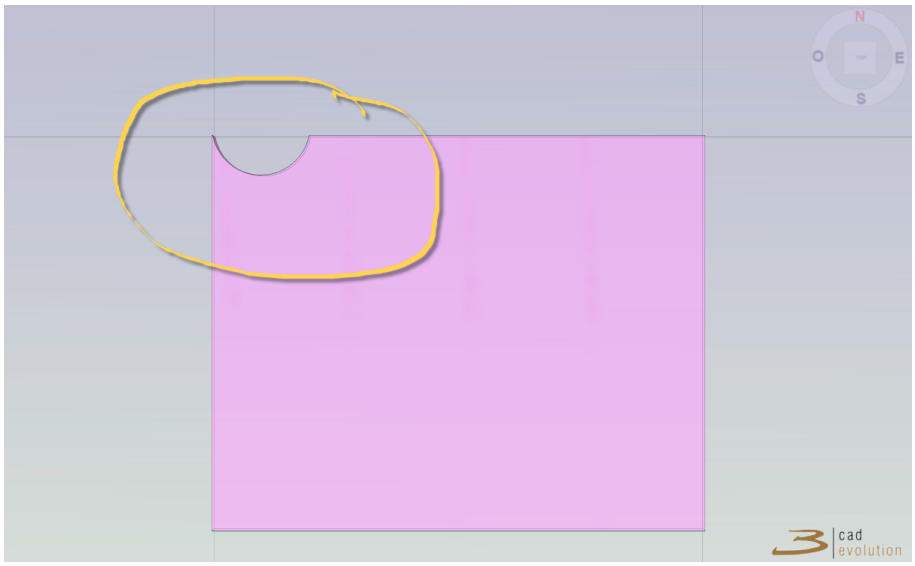


Figura 4.23: Sagoma con la fresa circolare semplice.

> **!** Il punto (0,0) del rettangolo coincide con il punto (0,0) della sagoma.

Esempio 45. Frese sovrapposte.

Le frese possono essere sovrapposte, Figura 4.23 codice 4.24.

Codice 137. Due frese sovrapposte.

```
!SAGOMA sag1 C40,0,0,50
!SAGOMA TM2 C40,0,0,100

!FRESA sagConFresa,500,400, \
      $(sag1);0;50;-10\
      $(sag2);0;120;0\
1,DIS=$(sagConFresa),A=10
```

Esempio 46. Comando *!FRESA*.

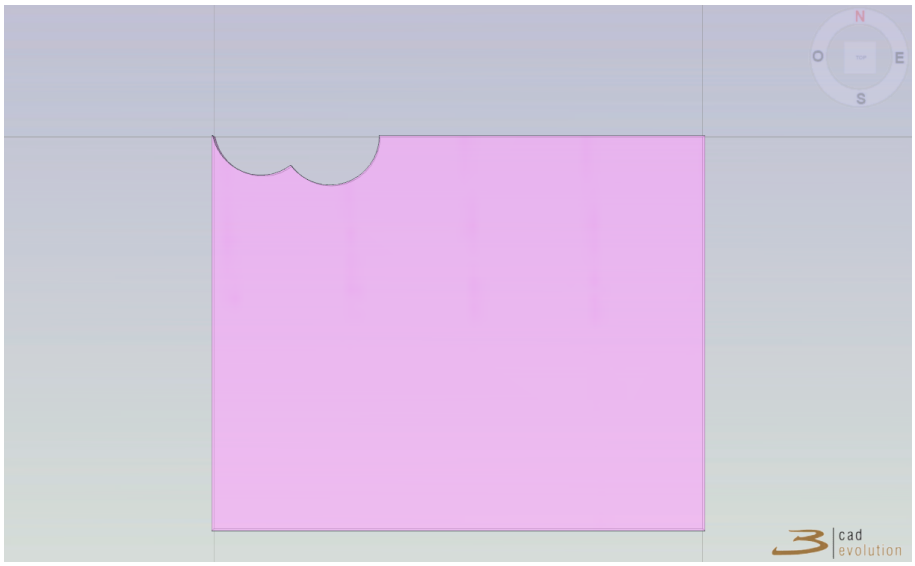


Figura 4.24: Due frese sovrapposte.

In questo esempio applichiamo la rotazione di 30 gradi alla fresa, Figura 4.25 nella pagina 157.

Codice 138. Utilizzo del comando !FRESA.

```
!SAGOMA TM1 0,0,50,0,20,50,0,50
!FRESA TMP,500,400, \
      $(TM1);0;120;-10;30\


1,DIS=$(sagConFresa),A=10
```

Esempio 47. Comando *!FRESA*.

Fresa semplice posizionata su quattro lati diversi, Figura 4.26 nella pagina 157.

Codice 139. Utilizzo del comando !FRESA.

```

!SAGOMA TM1 0,0,50,0,20,50,0,50
!FRESA TMP,500,400, \
      $(TM1);0;120;0,\
      $(TM1);1;120;0,\
      $(TM1);2;120;0,\
      $(TM1);3;120;0\


1,DIS=$(TMP),A=10
```

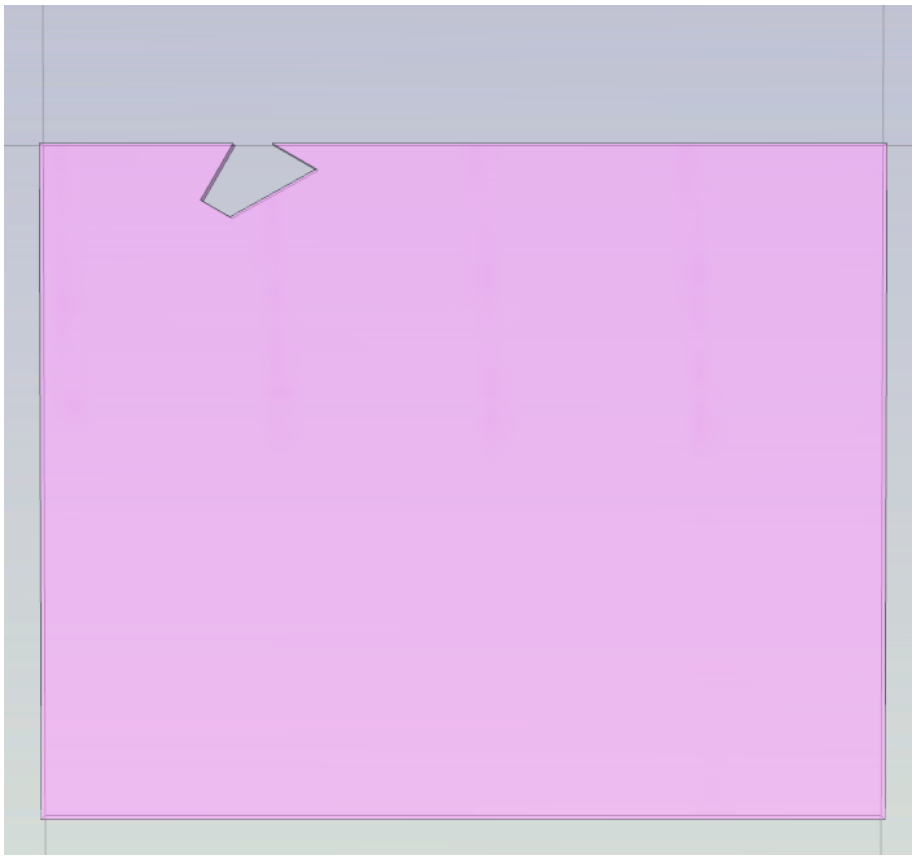


Figura 4.25: Fresa con utilizzo della rotazione.

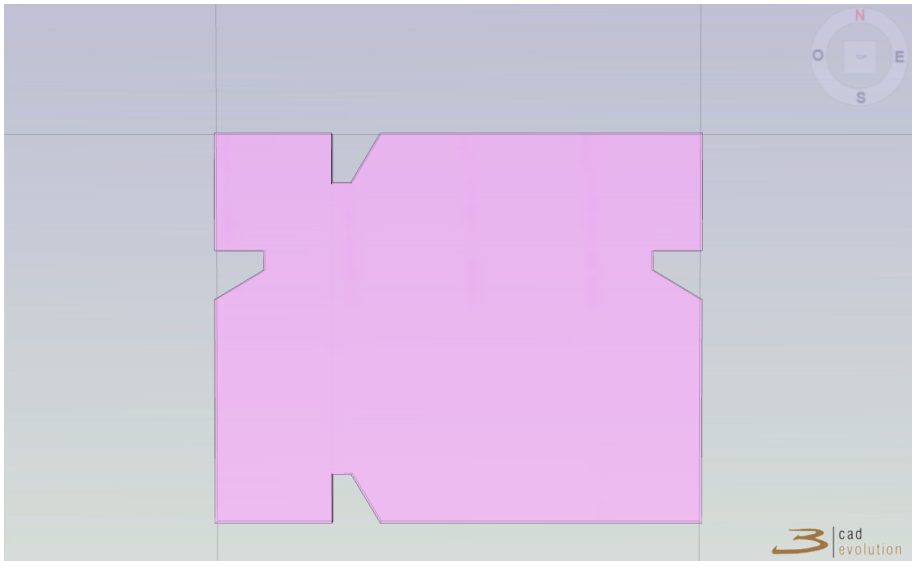


Figura 4.26: La stessa fresa su quattro lati diversi.

4.8.9 Comando *!LAMIERA*

Il comando *!LAMIERA* gestisce un caso particolare del comando sagoma in quanto basta creare un percorso e dichiarare uno spessore e viene creata una sagoma che si avvolge lungo tale percorso.

La sintassi da usare è la seguente:

!LAMIERA lamiera, spessore, percorso

Dove:

- lamiera: è la sagoma rappresentante la lamiera;
- spessore: è la larghezza della lamiera;
- percorso: è la sagoma rappresentate il percorso.

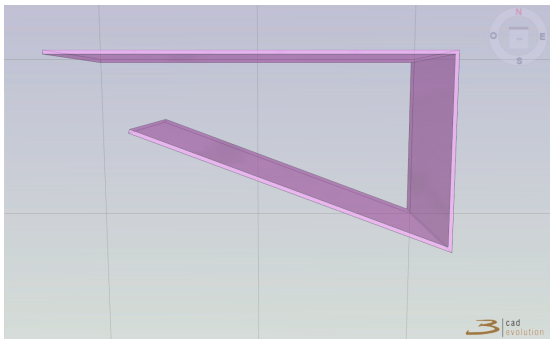


Figura 4.27: Lamiera con spessore 10.

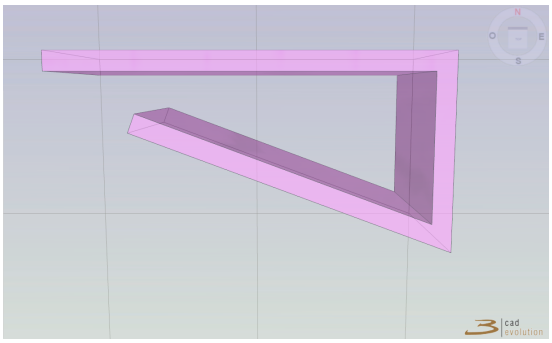


Figura 4.28: Lamiera con spessore 50.

Codice 140. Creare una lamiera.

```
1 !SAGOMA percorso 0,0 , 1000,0,1000,500, 200,200
2 !LAMIERA lamiera, 50, $(percorso)
3 001,DIS=$(lamiera)
```

4.8.10 Comando *!SAGMIRROR*

Il comando *!SAGMIRROR* specchia la sagoma in uno dei lati.

La sintassi da usare è la seguente:

!SAGMIRROR <newsag> oldsag <tipomirror: X-Y-XY>

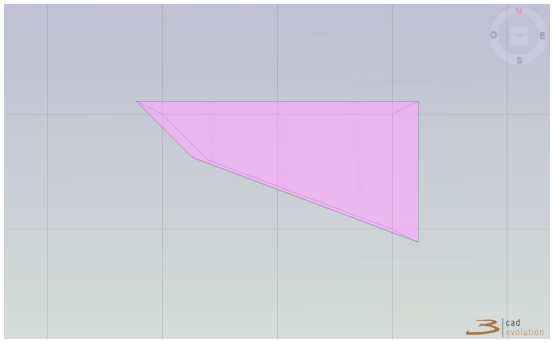


Figura 4.29: Sagoma di partenza, utilizzo comando !SAGMIRROR.

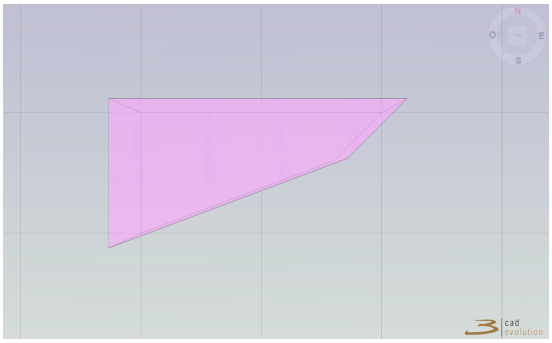


Figura 4.30: Sagoma specchiata nell'asse X.

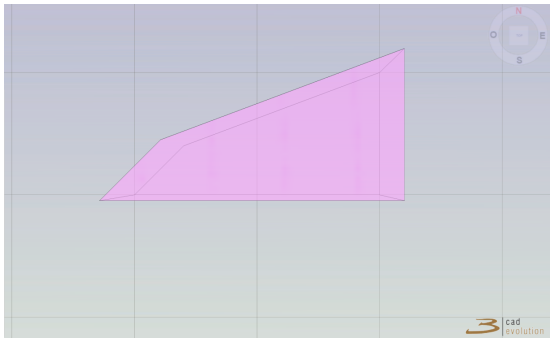


Figura 4.31: Sagoma specchiata nell'asse Y.

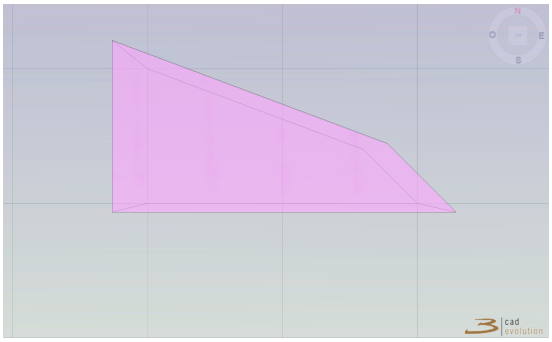


Figura 4.32: Sagoma specchiata nell'asse XY.

Crea una sagoma che specchia la sagoma di partenza, in asse X, Y o entrambi.

Esempio 48. Utilizzo comando *!SAGMIRROR*.

Codice 141. Specchiare la sagoma nell'asse X.

```
1 !SAGMIRROR TMP1 $(TMP) X
```

Specchia nell'asse X.

Codice 142. Specchiare la sagoma nell'asse Y.

```
1 !SAGMIRROR TMP1 $(TMP) X
```

Specchia nell'asse Y.

Codice 143. Specchiare la sagoma in entrambi gli assi.

```
1 !SAGMIRROR TMP1 $(TMP) XY
```

Specchia in entrambi gli assi.

4.8.11 Comandi *!SAGUNION, !SAGDIFF*

Il comando *!SAGUNION* unisce le due sagome : *sagoma1* e *sagoma2* e ritorna nella variabile *sagoma* la sagoma complessiva rappresentante la unione.

La sintassi da usare è la seguente:

!SAGUNION sagoma, sagoma1, sagoma2

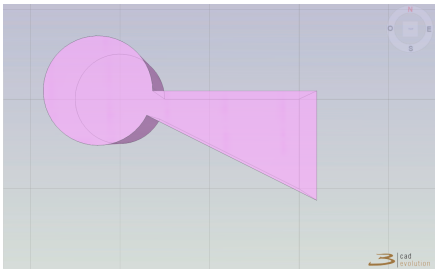


Figura 4.33: Unione di due sagome.

Codice 144. Utilizzo comando !SAGUNION(1).

```
1 !SAGOMA sagoma1 0,0 , 1000,0,1000,500
2 !SAGOMA sagoma2 C50, 0,0, 250
3 !SAGUNION sagoma , $(sagoma1), $(sagoma2)
4 001,DIS=$(sagoma)
```

Il comando *!SAGDIFF* sottrae la *sagoma2* dalla *sagoma1* e ritorna nella variabile *sagoma* la sagoma complessiva rappresentante la differenza.

La sintassi da usare è la seguente:

!SAGUNION sagoma, sagoma1, sagoma2

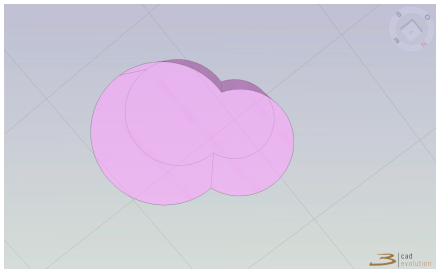


Figura 4.34: Unione di due sagome circolari.

Codice 145. Utilizzo comando !SAGUNION(2).

```
!SAGOMA sagoma1 C50,150,150 , 200
!SAGOMA sagoma2 C50, 0,0, 150
!SAGUNION sagoma , $(sagoma1), $(sagoma2)
001,DIS=$(sagoma)
```

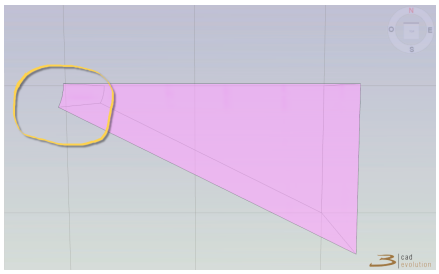


Figura 4.35: Differenza di due sagome.

Codice 146. Utilizzo comando !SAGDIFF(1).

```
!SAGOMA sagoma1 0,0 , 1000,0,1000,500
!SAGOMA sagoma2 C50, 0,0, 150
!SAGDIFF sagoma , $(sagoma1), $(sagoma2)
001,DIS=$(sagoma)
```

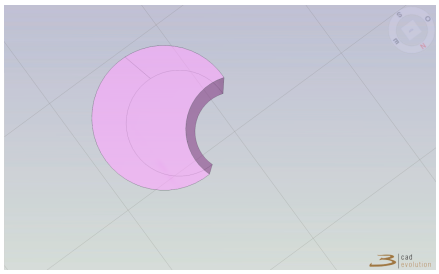


Figura 4.36: Differenza di due sagome circolari.

Codice 147. Utilizzo comando !SAGDIFF(2).

```
!SAGOMA sagoma1 C50,150,150 , 200
!SAGOMA sagoma2 C50, 0,0, 150
!SAGDIFF sagoma , $(sagoma1), $(sagoma2)
001,DIS=$(sagoma)
```

4.8.12 Comando *!SAGREL* o *!SAGSPOSTA*

Questi due comandi sono equivalenti spostano la sagoma e la ruotano di un angolo specifico.

La sintassi da usare è la seguente:

!SAGSPOSTA sagomaSpostata, sagoma, x, y, angolo

Trasforma la sagoma in un'altra sagoma, spostando tutti i punti di x,y e angolo di rotazione.

Esempio 49. Utilizzo comando *!SAGSPOSTA*.

Codice 148. Spostare la sagoma.

```
1 !SAGSPOSTA TMP1 $(tmp) 100 200 30
```

Questo comando sposta la sagoma lungo l'asse X di 100, lungo l'asse Y di 200 e la ruota di 30°.

Facciamo un esempio in cui è necessario utilizzare il comando *!SAGSPOSTA*.

Quando agli elementi si applicano dei materiali che hanno un senso di venatura, si rende necessario applicare delle rotazioni o degli spostamenti per seguire il senso corretto. Vediamo il risultato prodotto dal codice 149 in Figura 4.37 nella pagina 163. Si può notare che il pannello con il numero 2 ha il senso di venatura sbagliato perché il senso di venatura del pannello è orizzontale mentre il pannello è posizionato in verticale.

Codice 149. Due pannelli sagomati.

```
1   !SAGOMA tmp 0,0,1000,0,1000-400,400,0,400
2   1,dis=$(tmp),a=30,c1=2
3   !SAGOMA tmp 1000,0,1000,1000,1000-400,1000,1000-400,400
4   1,dis=$(tmp),a=30,c1=2
```

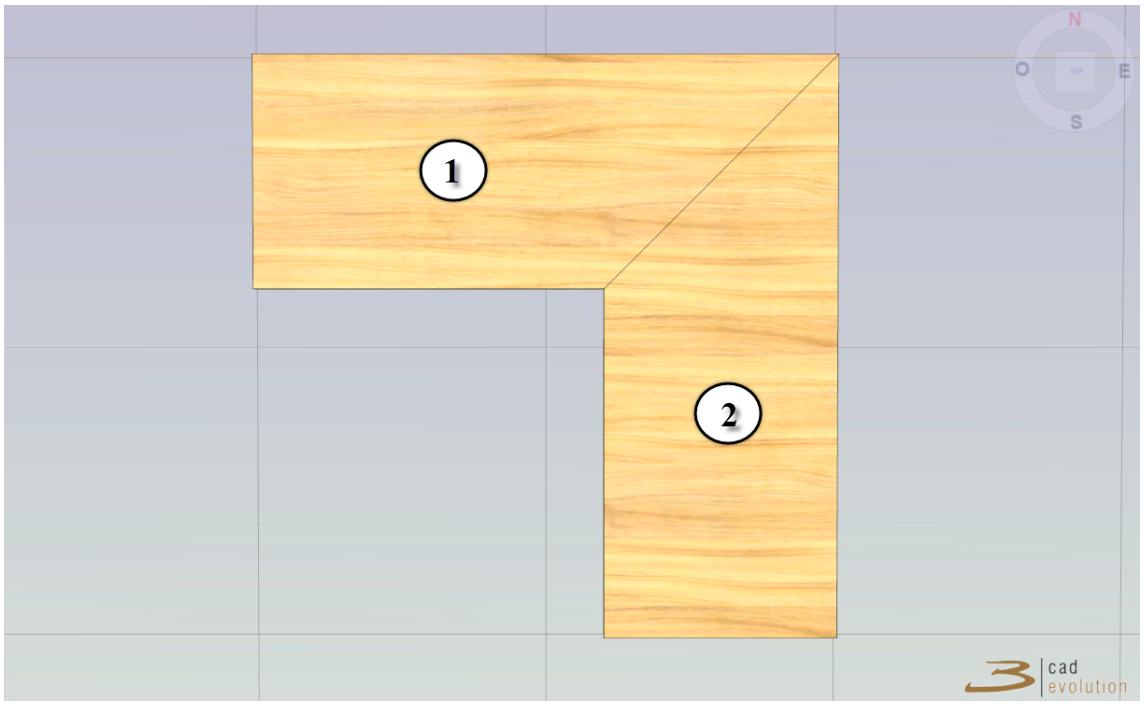


Figura 4.37: Un pannello orizzontale ed uno verticale con applicazione del materiale legno non corretta.

Proviamo a costruire una sagoma con il senso di venatura in orizzontale e poi girandola ottenere il secondo pannello, codice 150. In questo caso otteniamo i pannelli posizionati nel modo corretto, Figura 4.38.

Codice 150. Utilizzo del comando !SAGSPOSTA.

```
!SAGOMA tmp 0,0,1000,0,1000-400,400,0,400
 1,dis=$(tmp),a=30,c1=2
 !SAGOMA tmp 1000,0,1000,1000,1000-400,1000,1000-400,400

 !SAGSPOSTA tm2,$(tmp),1000,0,-90
 1,sl=1000,ay=-90,a=30,c1=2,dis=$(tm2)
```

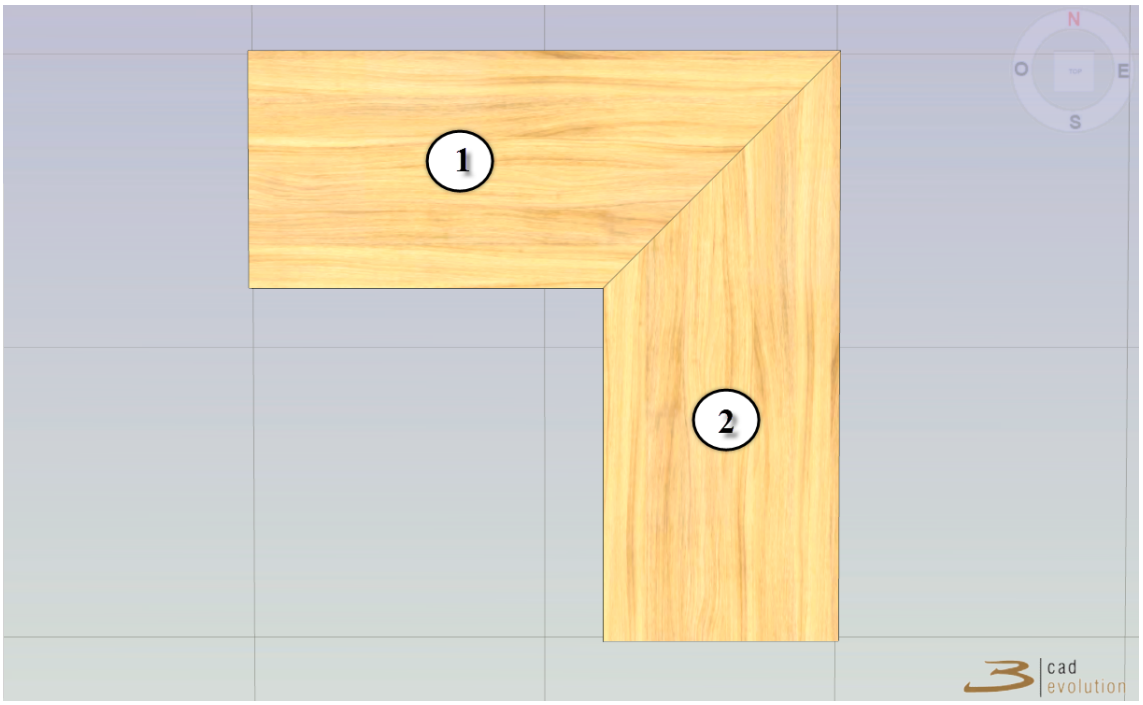


Figura 4.38: Un pannello orizzontale ed uno verticale con applicazione del materiale legno corretta.

4.8.13 Esempio complesso della gestione delle sagome

Per capire bene l'applicazione dei comandi di gestione delle sagome utilizzeremo un esempio complesso del mobile visibile in Figura 4.39 nella pagina 165.

Andiamo a costruire passo per passo il nostro mobile.

Partiamo dalla costruzione della sagoma principale. Definiamo un insieme di archi che andranno a comporre la sagoma, figura Figura 4.40 nella pagina 165. Per semplificare mettiamo i valori degli archi nelle variabili, codice 151.

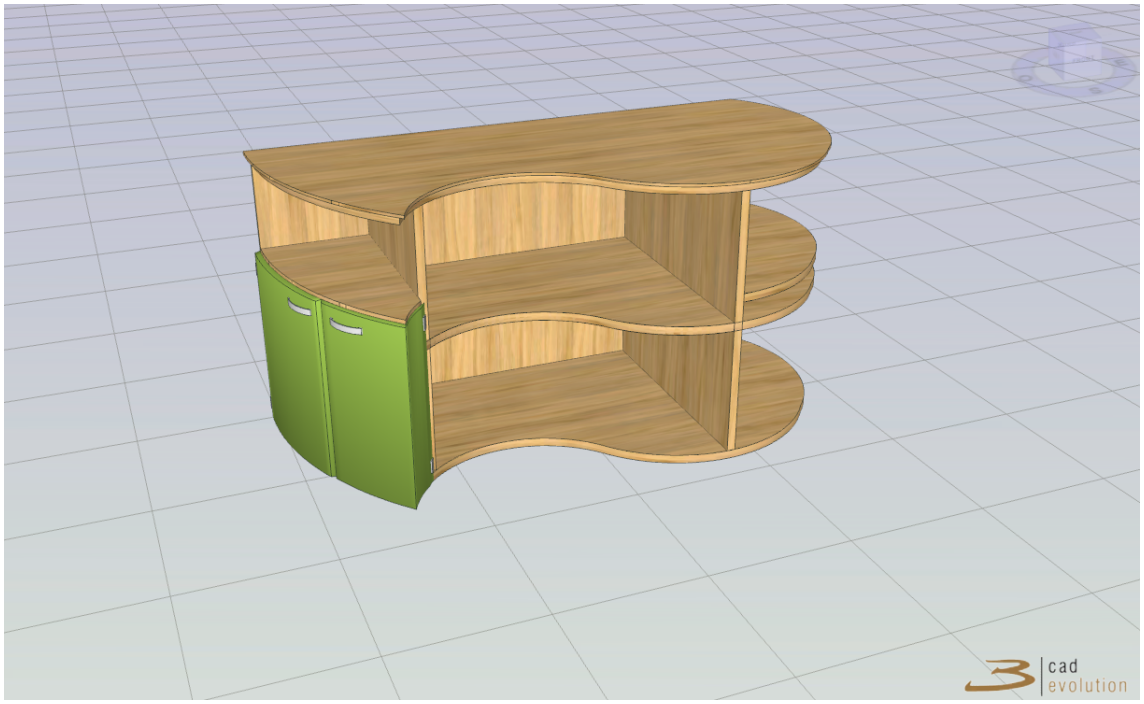


Figura 4.39: Mobile sagomato.

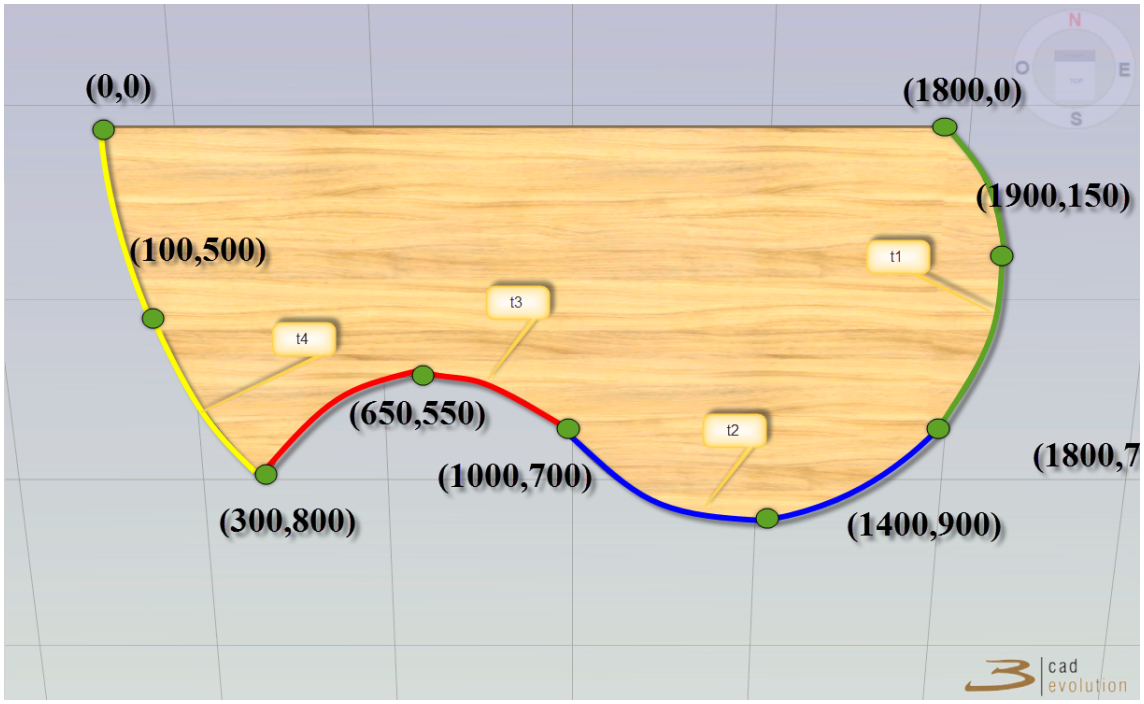


Figura 4.40: Gli archi della sagoma.

Codice 151. Definizione degli archi per la sagoma.

```
#LET l=2000,p=1000,a=1000
#LETS t1=A30,1800,0,1900,150,1800,700
#LETS t2=A30,1800,700,1400,900,1000,700
#LETS t3=A30,1000,700,650,550,300,800
#LETS t4=A30,300,800,100,500,0,0
```

Utilizzando il comando *!SAGOMA* andiamo a costruire la sagoma, figura Figura 4.41 codice 153.

Codice 152. Costruzione della sagoma

```
!SAGOMA tmp $(t1),$(t2),$(t3),$(t4)
503,SA=$A/2-$(_SPRIP:0)/2,SL=0,SP=0,DIS=$(TMP)
```

Adesso andiamo a costruire i fianchi come dal codice 153 , figura Figura 4.41.

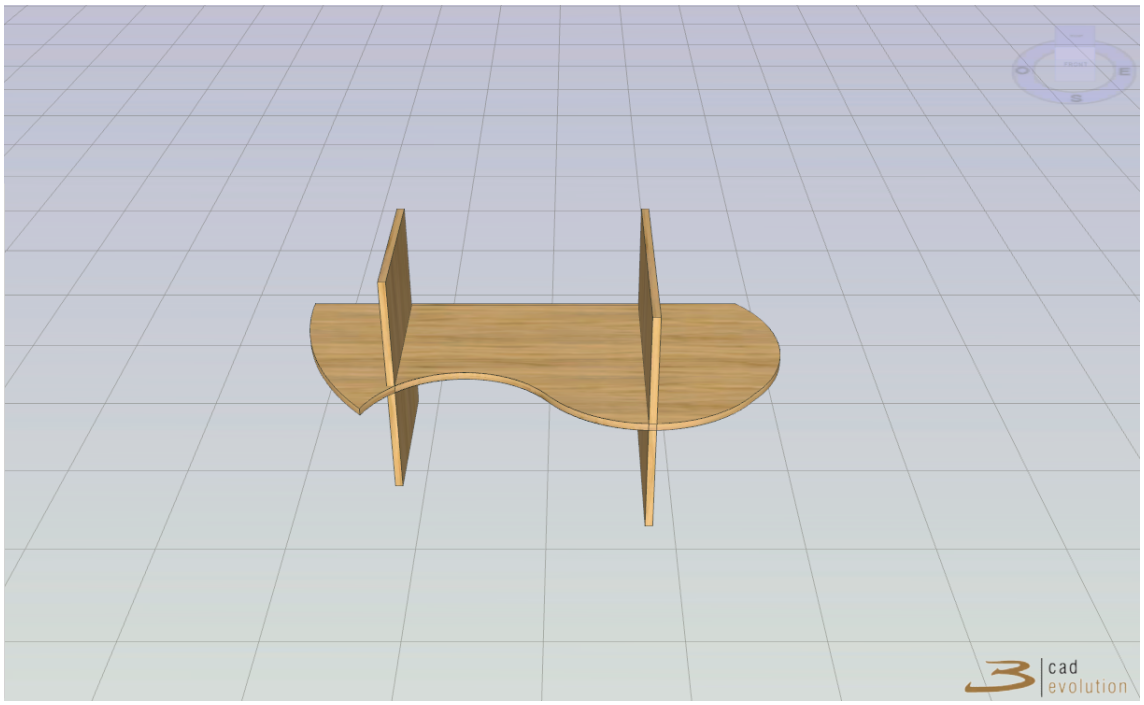


Figura 4.41: I fianchi sagomati.

Codice 153. Utilizzo del comando !SAGPERCX per la costruzione dei fianchi sagomati.

```
#LET VANOP1=400
#LET VANOP2=1400

// CALCOLO IL RIPIANO E LA SUA SAGOMA E POSIZIONO I FIANCHI
   DI STRUTTURA!
!SAGPERCX h1,$(VANOP1)+$(_SPFIA:0)/2,$(tmp)
!SAGPERCX h2,$(VANOP2)-$(_SPFIA:0)/2,$(tmp)

```

```
8
9
10 !SAGMID tx,$(h1),$(h2),$(tmp)
11 !SAGXY px1,py1,1,$(tX)
12 !SAGXY px0,py0,0,$(tX)
13
14 !SAGDUMP DM $(TX)
15 !SAGOMA TMM $(PX1),15,$(PX0),15,$(DM)
16 503,SA=$A/2-$(_SPRIP:0)/2,SL=0,SP=0,DIS=$(TMM)
17 501,sl=$(PX1)-$(_SPFIA),sa=$(_sprip:0),sp=15,a=$a-$(_sprip:0)
       *2,p=$(PY1)-15
18 501,sl=$(PX0),sa=$(_sprip:0),sp=15,a=$a-$(_sprip:0)*2,p=$(PY0
       )-15
```

Per poter posizionare i fianchi alla distanza specificata dalle variabili VANOP1 e VANOP2 utilizziamo il comando *!SAGPERCX* che restituisce il valore in percentuale rispetto la lunghezza del mobile. Il comando *!SAGMID* restituisce la sagoma visibile in figura Figura 4.42 rappresentata dalla linea gialla. Come si può vedere la sagoma TX restituita dal comando *!SAGMID* rappresenta solo una parte del ripiano. Per costruire il ripiano andiamo a prendere i valori sull'asse x del punto iniziale, riga 7 del codice 153 ed il punto finale, riga 8 del codice 153 della sagoma TX. Nella riga 9 del codice andiamo a prendere i punti che costituiscono la sagoma TX necessari alla costruzione della nuova sagoma rappresentante il ripiano. Prendendo i punti restituiti dal comando *!SAGDUMP* e cambiando il valore y dei punti p1 e p0 costruiamo la sagoma TMM rappresentante il ripiano, riga 10 del codice.

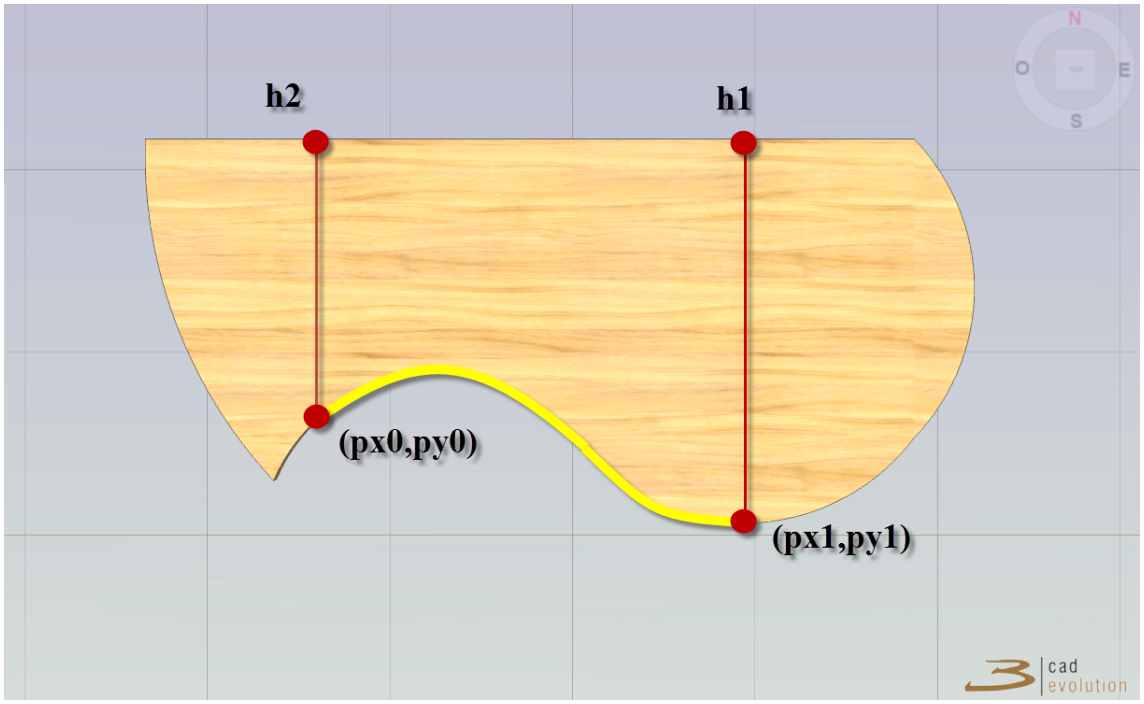


Figura 4.42: Applicazione del comando !SAGMID.

Adesso utilizzeremo i comandi *!SAGCLIPX* ed *!SAGCLIPY* nel codice 154 per la realizzazione di una sagoma rappresentante il ripiano, figura Figura 4.43 nella pagina 168.

Codice 154. Utilizzo dei comandi !SAGCLIPX e !SAGCLIPY per la costruzione dei ripiani sagomati.

```
1 !SAGCLIPX tmm,$(tmp),{$(VANOP2)+$(_SPFIA:0)/2},3000
2 !SAGCLIPY tmm,$(tmm),20,2000
3 503,SA=$A/2-$(_SPRIP:0)/2+100,SL=0,SP=0,DIS=$(TMM)
```

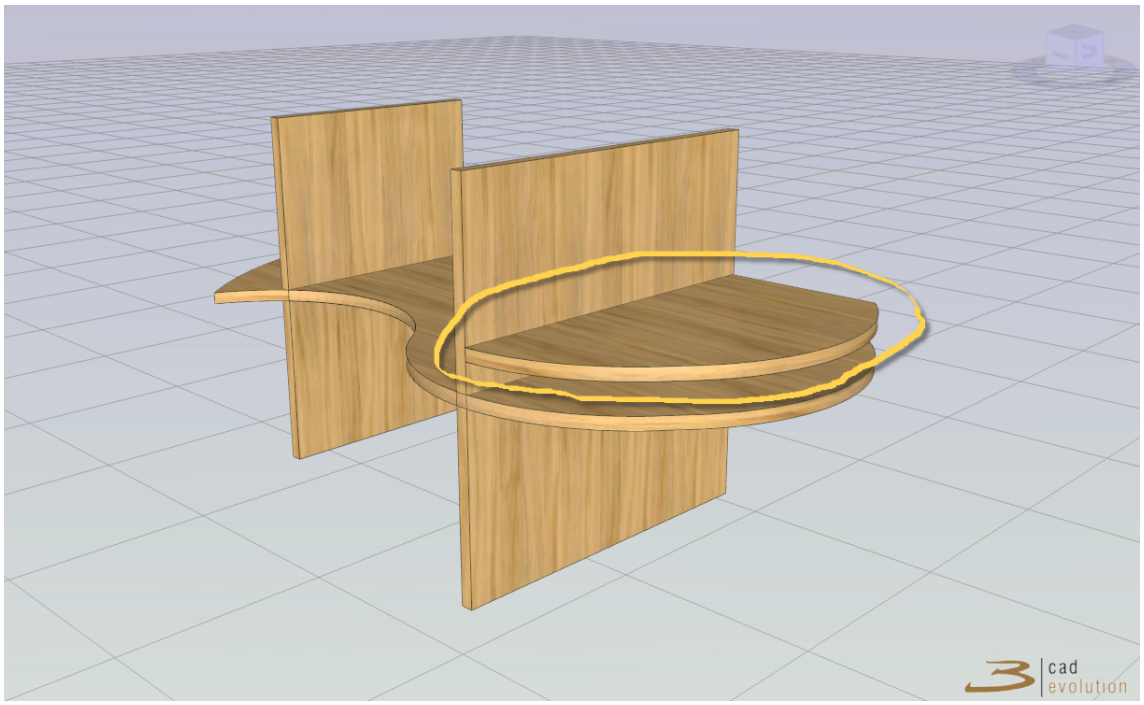


Figura 4.43: Ripiano sagomato.

Costruiamo il secondo ripiano a sinistra con gli stessi comandi, Figura 4.44 codice 155.

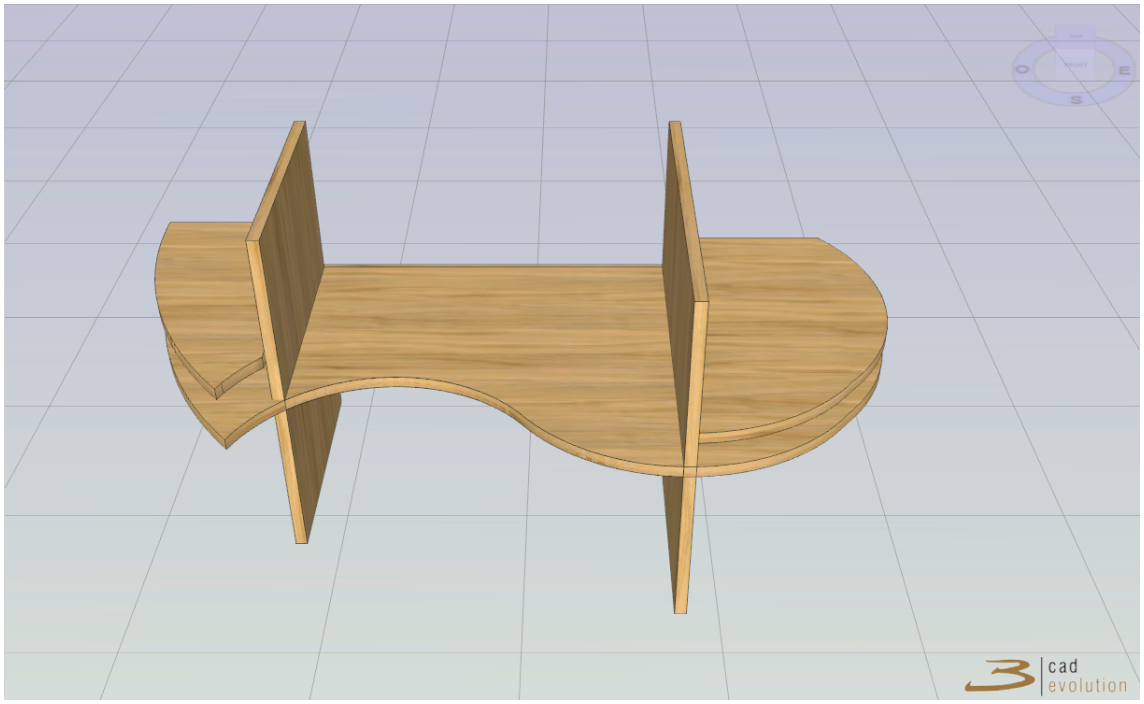


Figura 4.44: Due ripiani sagomati.

Codice 155. Utilizzo dei comandi !SAGCLIPX e !SAGCLIPY.

```
!SAGCLIPX tmm,$(tmp),{$(VANOP2)+$(_SPFIA:0)/2},3000
!SAGCLIPY tmm,$(tmm),20,2000
503,SA=$A/2-$(_SPRIP:0)/2+100,SL=0,SP=0,DIS=$(TMM)


!SAGCLIPX tmm,$(tmp),-100,{$(VANOP1)-$(_SPFIA:0)/2}
!SAGCLIPY tmm,$(tmm),20,2000
503,SA=650-$(_SPRIP:0)/2,SL=0,SP=0,DIS=$(TMM)
```

Come possiamo vedere dal codice 155, nella seconda e quinta riga abbiamo tagliato la nostra sagoma anche rispetto l'asse y. Questo è necessario nel nostro caso perché le due sub sagome rappresentano i ripiani e questo taglio è necessario per il successivo inserimento dello schienale. In Figura 4.45 nella pagina 170 è possibile vedere il ripiano senza il taglio in y.

Nel codice 156 andiamo ad inserire la base riga 1, cappello riga 2, schiena riga 3. Il risultato è visibile in Figura 4.46 nella pagina 170 .

Codice 156. Inserimento della base, cappello e schiena del mobile sagomato.

```
// BASE, CAPPELLO, TOP E SCHIENA
503,sl=0,sa=0,l=$l,a=30,p=$p,dis=$(tmp)
503,sl=0,sa=$a-$(_sprip:0),l=$l,a=30,p=$p,dis=$(tmp)
505,sa=$(_sprip:0),a=$a-$(_sprip:0)*2,l=1800
```

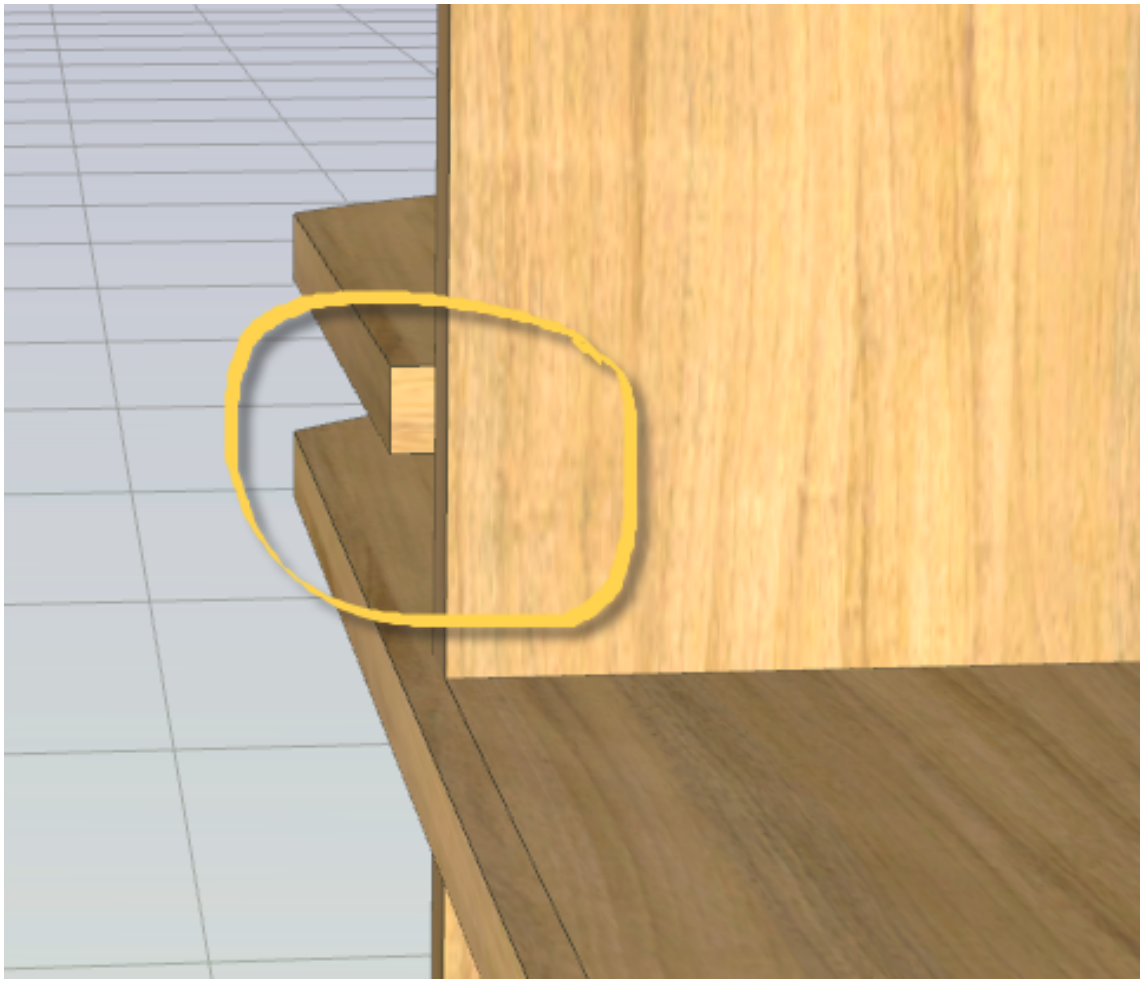


Figura 4.45: Necessità dell'utilizzo del comando !SAGCLIPY.

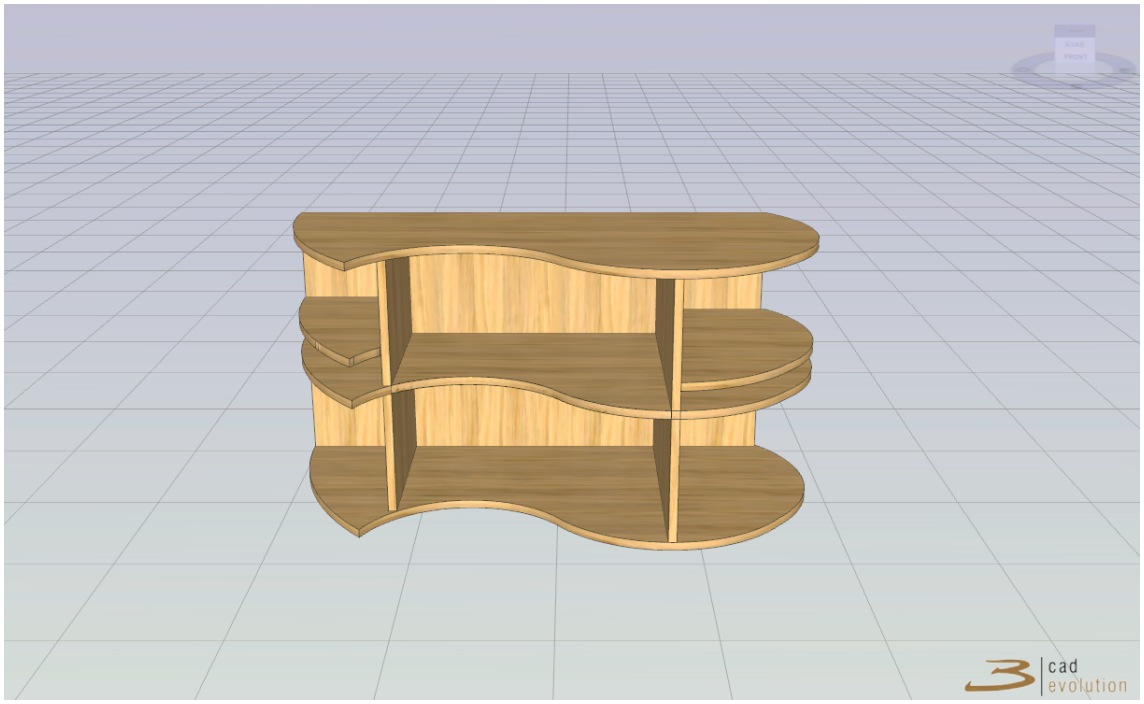


Figura 4.46: Inserimento della base, cappello e schienale del mobile.

Adesso andiamo a lavorare il top del mobile, codice 157.

Codice 157. Utilizzo del comando !SAGOFFSET per la costruzione del top.

```
!SAGOFFSET tmp1 $(tmp) -20
503,sa=$a,a=10,dis=$(tmp1)
```

Come è possibile notare dal codice stiamo utilizzando il comando *!SAGOFFSET* per costruire l'effetto visibile in Figura 4.47. Abbiamo utilizzato il valore dell'offset negativo per far sporgere il top del mobile.

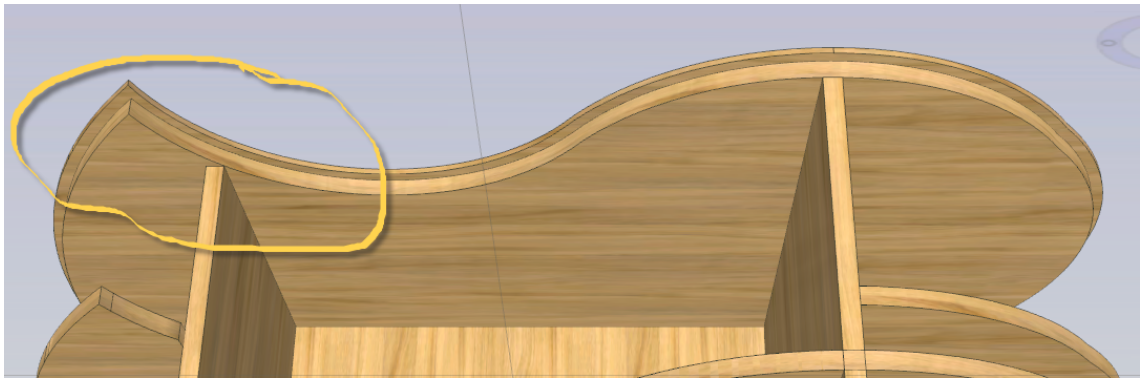


Figura 4.47: Top sporgente.

Per togliere la parte del top dove è presente la schiena del mobile possiamo utilizzare il comando *!SAGCLIPY*, codice 158.

Codice 158. Taglio del bordo posteriore del top.

```
!SAGOFFSET tmp1 $(tmp) -20
!SAGCLIPY tmp2,$(tmp1),0,3000
503,sa=$a,a=10,dis=$(tmp2)
```

Successivamente andiamo a costruire le ante che in questo caso necessitano anche di essere modellate rispetto alla sagoma. Il codice 159 crea l'anta visibile in Figura 4.48 nella pagina 172.

Codice 159. Costruzione dell'anta sagomata.

```
!SAGPERCX h1,$(VANOP1),$(tmp)
!SAGPERCX h2,$(VANOP2),$(tmp)


// A SX
#LET H0=(1+$(H1))/2
#LETS nota=Anta Speciale Sagomata

!SAGMID tx,$(H0),1,$(tmp)
!SAGALIGN tmp5,pp,1,$(tx)
515,SL=$(pp:0),SP=$(pp:1) ,a=650,L=$(pp:3),AY=-$(pp:2),GL=$(.
    percorso)\3ds\script\test.xp,X1=$(tmp5),x6=$(nota)
```

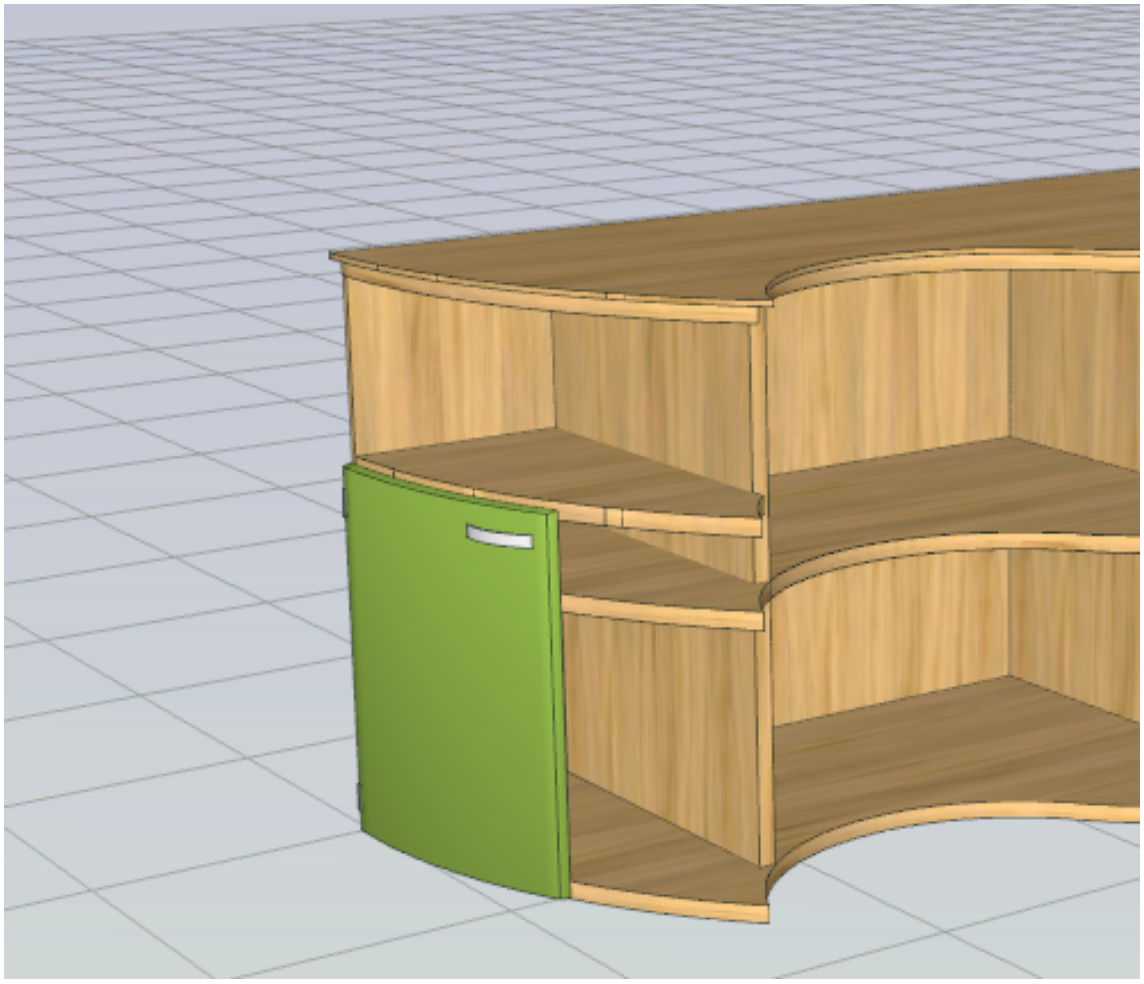


Figura 4.48: Anta sagomata.

Come si può vedere una volta individuata la sub sagoma con la forma della quale vogliamo modellare l'anta è necessario allineare la sagoma rispetto l'asse x con il comando *!SAGALIGN* per metterla nella posizione corretta. I valori restituiti dal comando *!SAGALIGN* vengono passati come valori di spostamento nella tipologia che rappresenta l'anta.

4.9 Comandi matematici delle regole

Questi comandi si basano su funzioni che richiedono 2 box fra loro collegati e la relazione esistente tra punti di box1 e box2, e l'intersezione di linee virtuali definite da box1 e box2. Sono usati principalmente nelle regole degli elementi angolari.

4.9.1 Comando *!MTCOLLEGA*

Gestisce un collegamento con il box precedente di tipo 198. Il collegamento prevede i seguenti parametri:

La sintassi da usare è la seguente:

!MTCOLLEGA (PXCOLL), (PYCOLL), (vPX), (PY), (ANGOLO)

- PXCOLL, PYCOLL: posizione x e y del box collegato (riferite a se stesso);
- PX, PY: punto di contatto con il box corrente;
- ANGOLO: angolo di rotazione.

Il sistema mette a contatto pxcoll, pycoll con px, py e ruota di angolo.

4.9.2 Comando *!DISTANZA*

Inserisce in variabile la distanza tra 2 punti

La sintassi da usare è la seguente:

!DISTANZA (VARIABILE), (PX1), (PY1), (PX2), (PY2)

4.9.3 Comando *!RETTA*

Calcola una retta a cui assegna un numero identificativo (da 0 a 9) parallela (tipo=0) o perpendicolare (tipo=1) al box selezionato a una distanza (distanza) dall'origine.

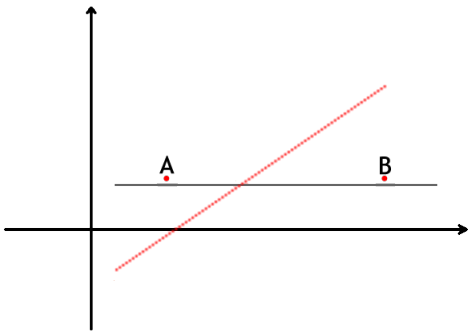


Figura 4.49: Retta (1).

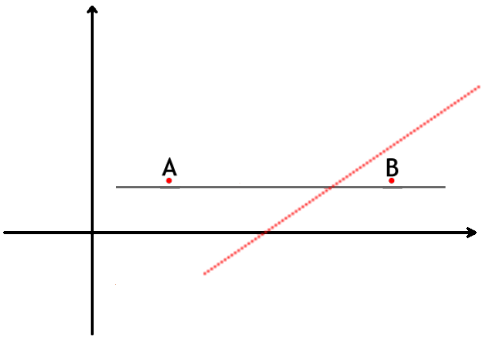


Figura 4.50: Retta (2).

La sintassi da usare è la seguente:

!RETTA (ID),(TIPO),(DISTANZA),[BOXPRECEDENTE]

La retta può essere riferita al box collegato con il parametro [boxprecedente]=1 .

4.9.4 Comando *!RETTA2*

La sintassi da usare è la seguente:

!RETTA2 R, PX1, PY1, PX2, PY2, DISTANZA, ANGOLO

- R: è il numero della retta che viene creata
- PX1, PY1, PX2, PY2: sono i due punti per cui passa una retta
- DISTANZA: è la distanza di cui viene spostata la retta dai due punti !RETTA2 1, 20, 20, 100, 20, 30, 40

- ANGOLO: (facoltativo) è l'angolo di inclinazione della nuova retta Se Angolo è impostato e distanza ha valore 0 si ottiene una retta passante per il punto PX1, PY1 e con angolo ANGOLO rispetto alla retta che passa per PX1, PY1 e PX2, PY2.

4.9.5 Comando *!MATHAZZERA*

Ricalcola i collegamenti dei box e azzera ogni linea tracciata di riferimento.

La sintassi da usare è la seguente:

!MATHAZZERA

4.9.6 Comando *!INTERSECA*

Calcola il punto di intersezione tra 2 rette identificate da idretta1 e idretta2, inserendolo nelle variabili varx e vary e indicando in errore il valore 1 se l'intersezione ha avuto successo.

La sintassi da usare è la seguente:

!INTERSECA (VARX), (VARY), (IDRETTA1), (IDRETTA2), [ERRORE]

4.9.7 Comando *!PUNTO*

Trasforma il punto (puntox, puntoy) del box collegato in varx e vary del box corrente.

La sintassi da usare è la seguente:

(VARX), (VARY), (PUNTOX), (PUNTOY) !PUNTO (VARX),(VARY), (PUNTOX), (PUNTOY)

4.9.8 Comando *!PUNTOL*

Trasforma il punto (puntox, puntoy) del box corrente in varx e vary del box collegato.

La sintassi da usare è la seguente:

!PUNTOL (VARX), (VARY), (PUNTOX), (PUNTOY)

4.9.9 Comando *!PUNTOREL*

Trasforma il punto (puntox, puntoy) del box corrente in varx e vary di un nuovo sistema di coordinate determinato da (originex, originey), origine del nuovo sistema di coordinate, e da (ang), angolo di rotazione rispetto al sistema di riferimento del box corrente.

La sintassi da usare è la seguente:

!PUNTOREL (VARX), (VARY), (ORIGINEX), (ORIGINEX), (ANG), (PUNTOX), (PUNTOY)

4.10 Comandi per il disegno 2D

I comandi per il disegno 2D si suddividono i 3 categorie: comandi che disegnano solo i contorni di una forma, comandi che disegnano forme piene e istruzioni complete.

4.10.1 Comando *!!RECT*

Questo comando permette di disegnare un rettangolo con linee di contorno.

La sintassi da usare è la seguente:

```
!!RECT x, y, w, h, [color], [penStyle], [penw]
```

Dove:

- X e Y: sono le coordinate iniziali;
- W: è la larghezza;
- H: è l'altezza;
- Color: è il colore RGB (no alfa);
- PenStyle: stile della penna che può assumere i seguenti valori:
  - 0 – Solid
  - 1 – Dash
  - 2 – Dot
  - 3 – DashDot
- PenW: è lo spessore.

4.10.2 Comando *!!LINE*

Permette di disegnare una linea.

La sintassi da usare è la seguente:

```
!!LINE x, y, h, [color], [penStyle], [penw]
```

4.10.3 Comando *!!POLY*

Disegna una polilinea.

La sintassi da usare è la seguente:

```
!!POLY [color], x, y, x1, y1, x2, y2 ...
```

4.10.4 Comando *!!CIRCLE*

Disegna un cerchio con solo linee di contorno.

La sintassi da usare è la seguente:

```
!!CIRCLE x ,y, r, [color], [penStyle],[penw]
```

4.10.5 Comando *!!ARC*

Disegna un arco.

La sintassi da usare è la seguente:

```
!!ARC x, y, r, startAngle, endAngle, [color], [penStyle], [penW]
```

Dove:

- X e Y: sono le coordinate iniziali;
- R: è il raggio;
- StartAngle: inizio angolo;
- EndAngle: fine angolo.

4.10.6 Comando *!!PIE*

Questo comando permette di disegnare un arco pieno.

La sintassi da usare è la seguente:

```
!!PIE x , y, r, startAngle, endAngle, [color]
```

4.10.7 Comando *!!FILLRECT*

Disegna un rettangolo pieno.

La sintassi da usare è la seguente:

```
!!FILLRECT x ,y, w, h[color]
```

4.10.8 Comando *!!FILLCIRCLE*

Disegna un cerchio pieno.

La sintassi da usare è la seguente:

!!FILLCIRCLE x, y, w, h[color]

4.10.9 Comando *!!FILLPOLY*

Disegna una polilinea piena.

La sintassi da usare è la seguente:

!!FILLPOLY color, x, y ,x1, y1, x2, y2 ...

4.10.10 Comando *!!STRING*

Questo comando permette di scrivere un testo a video.

La sintassi da usare è la seguente:

!!STRING x, y, text, [angle], [color]

4.10.11 Comando *!!PUSHFONT*

Crea un font e lo imposta come attivo.

La sintassi da usare è la seguente:

!!PUSHFONT nomefont, size, bolt, italic

4.10.12 Comando *!!POPFONT*

Scarica il font attivo.

La sintassi da usare è la seguente:

!!POPFONT

4.10.13 Comando *!!PUSHMATRIX*

Copia in memoria la matrice di trasformazione corrente.

La sintassi da usare è la seguente:

```
!!PUSHMATRIX
```

4.10.14 Comando *!!POPMATRIX*

Recupera la matrice di trasformazione salvata in precedenza.

La sintassi da usare è la seguente:

```
!!POPMATRIX
```

4.10.15 Comando *!!TRASLA*

Aggiunge alla matrice di trasformazione un'operazione traslazione.

La sintassi da usare è la seguente:

```
!!TRASLA x, y, z
```

4.10.16 Comando *!!RUOTA ANGOLO*

Aggiunge alla matrice di trasformazione un'operazione rotazione.

La sintassi da usare è la seguente:

```
!!RUOTA ANGOLO
```

4.10.17 Comando *!!LINETO*

Traccia una linea da punto impostato precedentemente al punto definito.

La sintassi da usare è la seguente:

```
!!LINETO x, y, [color], [penStyle], [penw]
```

4.10.18 Comando *!!MOVETO*

Imposta il punto corrente.

La sintassi da usare è la seguente:

!!MOVETO x, y

4.10.19 Comando *!!DXF*

Introduce file DXF.

La sintassi da usare è la seguente:

!!DXF x, y, zoom, NomeFile

Esempio 50. Utilizzo comando *!!DXF*

Codice 160. Introdurre file DXF.

```
#LET Scala=25
!!DXF 1,1,$(Scala),$(.percorso)\foto\dxf\file.dxf}
```

4.10.20 Comando *!!DXF2*

Introduce file DXF.

La sintassi da usare è la seguente:

!!DXF2 x, y, scalaX, scalaY, NomeFile

4.10.21 Comando *!!BRECT*

Inserisce un'icona.

La sintassi da usare è la seguente:

!!BRECT x, y, w, h, comando, TipiIcona, angolo, colore, des

Dove:

- Comando:
- Tipi Icona:
- Angolo
- Colore:
- Des:

4.11 Funzioni

Al fine di facilitare l'accesso a dati e controlli particolari all'interno delle regole, sono state introdotte delle funzioni.

La sintassi da usare è la seguente:
$[<nomefunzione>; par1; par2; ...; parN]

Dove par1, ..., parN sono parametri che variano a seconda della funzione.

```
1 $[FORMAT; $a; 000]
```

Le funzioni trovano ampio utilizzo anche all'interno delle stampe parametriche e nelle elaborazioni dei dati anagrafici. Cerchiamo di spiegarne l'utilizzo e la sintassi di seguito.

4.11.1 Funzione LOOK

La funzione LOOK trova, all'interno di una variante, l'elemento che è maggiore o uguale rispetto al codice inserito e restituisce l'intera riga sotto forma di vettore.

La sintassi da usare è la seguente:
$[LOOK; <nomevariante>; <codice>]

Esempio 51. utilizzo funzione LOOK
si supponga che in una tabella misure siano riportati nelle righe di una colonna codice i seguenti valori: 100,200,500,1000,1500; l'istruzione, assegna alla variabile misura il vettore avente come componenti i campi della riga avente codice 1000, poiché è il valore immediatamente superiore a 800.

Codice 161. Funzione LOOK.

```
1 #LETS misura=$[LOOK; misure; 800]
```

4.11.2 Funzione LOOKB

La funzione LOOKB trova, all'interno di una variante, l'elemento che è minore o uguale rispetto al codice inserito e restituisce l'intera riga sotto forma di vettore.

La sintassi da usare è la seguente:
$[LOOKB; <nomevariante>; <codice>]

Esempio 52. Utilizzo funzione LOOKB.
si supponga che in una tabella misure siano riportati nelle righe di una colonna codice i seguenti valori: 100,200,500,1000,1500; l'istruzione assegna alla variabile misura il vettore avente come componenti i campi della riga avente codice 500, poiché è il valore immediatamente inferiore a 800.

Codice 162. Funzione LOOKB.

```
1 #LETS misura=$[LOOKB; misure; 800]
```

4.11.3 Funzione IIF

La funzione IIF si comporta come un IF in linea, vale a dire che se la condizione è vera viene accettato il primo valore, mentre se è falsa viene accettato il secondo. In questo esempio viene assegnato alla variabile PIPPO il valore ALFA se la variabile A è maggiore di 300, il valore BETA altrimenti.

La sintassi da usare è la seguente:
$[IIF; <condizione>; valore1; valore2]

Esempio 53. Utilizzo funzione IIF.

Codice 163. IF in linea.

```
1 #LETS PIPPO=$[IIF; $A>300; ALFA; BETA]
```

4.11.4 Funzione CASE

La funzione CASE valuta il contenuto di un'espressione e, a seconda del risultato, restituisce il valore corrispondente, avendo a disposizione varie possibilità.

La sintassi da usare è la seguente:
$[CASE; <condizione>; valore1; valore2; ...; valoreN]

Esempio 54. Utilizzo funzione CASE
In questo esempio viene assegnato alla variabile PIPPO il valore PRIMO se la variabile A è uguale a 1, il valore SECONDO se è uguale a 2 e così via. L'assegnazione è di tipo posizionale, di conseguenza la variabile $A è di tipo numerico e può assumere valori interi da uno a n.

Codice 164. Funzione CASE.

```
#LETS PIPPO=$[CASE; $A; PRIMO; SECONDO; TERZO ...]
```

4.11.5 Funzione DIR

La funzione DIR restituisce il valore 1 se esiste il file richiesto, il valore 0 in caso contrario.

La sintassi da usare è la seguente:
$[DIR; <nomefile>]

Esempio 55. Utilizzo funzione DIR
in PIPPO sarà contenuto il valore 1 se il file in oggetto è stato trovato e 0 se non c'è

Codice 165. Funzione DIR.

```
#LETS PIPPO=$[DIR; $(.percorso)\gene.erx]
```

4.11.6 Funzione LEN

La funzione LEN restituisce la lunghezza di una stringa assegnata.

La sintassi da usare è la seguente:
$[LEN; stringa]

Esempio 56. Utilizzo funzione LEN
In questo esempio viene assegnato alla variabile PIPPO il valore 4, poiché la stringa ALFA é composta da 4 caratteri; si noti che la stringa da valutare è stata scritta senza apici. Al posto della stringa ci può essere una variabile, es. $(alfa).

Codice 166. Funzione LEN.

```
#LETS PIPPO=$[LEN; ALFA]
```

4.11.7 Funzione MID

La funzione MID ritorna una sottostringa della stringa selezionata.

La sintassi da usare è la seguente:
$[MID; <parola>; <da carattere>; <numero caratteri>]

Esempio 57. Utilizzo funzione MID completa
In questo esempio viene assegnato alla variabile PIPPO il valore RO.

Codice 167. Funzione MID.

```
#LETS PIPPO=$[MID; PAROLA; 3; 2]
```

Esempio 58. Utilizzo funzione MID incompleta
In questo esempio viene assegnato alla variabile PIPPO il valore ROLA, perché, se non viene specificato il numero di caratteri, la funzione MID restituisce la parte finale della stringa, a partire dal carattere specificato.

Codice 168. Funzione MID incompleta.

```
#LETS PIPPO=$[MID; PAROLA; 3]
```

4.11.8 Utilizzo funzione ESTRAI o EX

La funzione ESTRAI restituisce una sottostringa della stringa selezionata delimitata a destra da un separatore. Il separatore può essere costituito da una sequenza di diversi caratteri.

La sintassi da usare è la seguente:
$[ESTRAI; <variabile>; <separatore>; <posizione>]

Esempio 59. Utilizzo funzione ESTRAI.
In questo esempio viene assegnato alla variabile PIPPO il valore Terzo. La stringa su cui si fa la ricerca può essere costituita da una variabile.

Codice 169. Funzione ESTRAI.

```
#LETS PIPPO=$[ESTRAI; Primo, Secondo, Terzo, Quarto; ,; 3]
```

4.11.9 Funzione INSTR

La funzione INSTR determina in che posizione si trova una certa stringa all'interno di un'altra stringa.

La sintassi da usare è la seguente:
$[INSTR; <stringa>; <stringa da cercare>]

Esempio 60. Utilizzo funzione INSTR
In questo esempio viene assegnato alla variabile PIPPO il valore 7, in quanto la parola cercata 'secondo' inizia nel settimo carattere della stringa 'Primo, Secondo, Terzo.

Codice 170. Funzione INSTR.

```
1 #LETS PIPPO=$[INSTR; Primo, Secondo ,Terzo; Secondo]
```

4.11.10 Funzione FORMAT

La funzione FORMAT formatta un campo secondo il formalismo della funzione Format$ di VBA.

La sintassi da usare è la seguente:
$[FORMAT; <stringa>; <formato>]

Esempio 61. Utilizzo funzione FORMAT (1).
In questo esempio viene assegnato alla variabile PIPPO il valore 0012.

Codice 171. Funzione FORMAT (1).

```
1 #LETS PIPPO=$[FORMAT; 12; 0000]
```

Esempio 62. Utilizzo funzione FORMAT (2)
In questo esempio viene assegnato alla variabile PIPPO il valore 13-06-12.

Codice 172. Funzione FORMAT (2).

```
1 #LETS PIPPO=$[FORMAT; 130612; 00-00-00]
```

4.11.11 Funzione FORMATV

La funzione FORMATV ha le stesse caratteristiche del FORMAT e serve per gestire la virgola.

La sintassi da usare è la seguente:
$[FORMAT; <stringa>; <formato>]

Esempio 63. Utilizzo funzione FORMATV
In questo esempio viene assegnato alla variabile PIPPO il valore 12,45.

Codice 173. Funzione FORMATV.

```
#LETS PIPPO=$[FORMATV; 12.45; 00.00]
```

4.11.12 Funzione SQL e SQLA

La funzione SQL esegue per l'appunto un comando SQL. Il comando è un comando di selezione che deve ritornare un campo che corrisponde al ritorno della funzione.

La sintassi da usare è la seguente:
$[SQL; <comando SQL>]

Esempio 64. Utilizzo funzione SQL

Codice 174. Funzione SQL.

```
#LETS NOME=$[SQL; SELECT rag1 from clienti where cod='0001']
```

La funzione SQLA è identica alla precedente, solo che va eseguita solamente sul database alternativo (presente in alcune procedure e contenente informazioni di tipo gestionale).

4.11.13 Funzione CODVAR

La funzione CODVAR ritorna la descrizione del codice variante impostato.

La sintassi da usare è la seguente:
$[CODVAR; <nomevariante>]

Esempio 65. Utilizzo funzione CODVAR
Alla variabile PIPPO viene assegnato il valore Colore Struttura, che è, appunto, la descrizione della variante di codice STR. La funzione equivale alla chiamata con SQL: $[SQL;SELECT DES FROM VARIANTI WHERE CODVAR='STR']

Codice 175. Funzione CODVAR.

```
#LETS PIPPO=$[CODVAR; STR]
```

4.11.14 Funzione VAR o VAROPZ

La funzione funzioneVAR o VAROPZ ritorna la descrizione dell'opzione scelta relativamente ad una determinata variante.

La sintassi da usare è la seguente:
$[VAR<formato><lingua>; <Num/Nome Variante; <elenco varianti>]

4.11.15 Funzione VARIANTE

La funzione VARIANTE ritorna la descrizione dell'opzione relativamente ad una variante secondo i vari formati elencati precedentemente, in relazione alla funzione VAROPZ.

La sintassi da usare è la seguente:
$[VARIANTE; <cod.variante = cod.opzione>; <formato>]

Esempio 66. Utilizzo funzione VARIANTE
Alla variabile PIPPO viene assegnata la stringa Tipo anta=telaio.

Codice 176. Funzione VARIANTE.

```
#LETS PIPPO=$[VARIANTE; anta=02; 1]
```

4.11.16 Funzione WEEK

Questa funzione restituisce il numero corrispondente della settimana a cui appartiene una certa data.

La sintassi da usare è la seguente:
$[WEEK; <gg/mm/aaaa>]

Esempio 67. Utilizzo unzione WEEK
In questo esempio viene assegnato alla variabile PIPPO il valore 11 che è il numero corrispondente alla settimana del 10 marzo 2003.

Codice 177. Funzione WEEK.

```
#LETS PIPPO=$[WEEK; <13/06/2013>]
```

4.11.17 Funzione DATA

Questa funzione incrementa la data definita di un certo numero di giorni.

La sintassi da usare è la seguente:
$[DATA; <gg/mm/aaaa>; n]

Esempio 68. Utilizzo funzione DATA.
In questo esempio viene assegnata alla variabile PIPPO la stringa contenente la seguente data: 13/07/2012.

Codice 178. Funzione DATA.

```
1 #LETS PIPPO=$[DATA; 13/06/2012; 30]
```

4.11.18 Funzione DIM

Questa funzione formatta le dimensioni secondo lo schema dim1Xdim2Xdim3.

La sintassi da usare è la seguente:
$[DIM; dim1; dim2; dim3]

Esempio 69. Utilizzo funzione DIM
In questo esempio viene assegnata alla variabile PIPPO la stringa: 120x720x340.

Codice 179. Funzione DIM.

```
1 #LETS PIPPO=$[DIM; 120; 720; 340]
```

4.11.19 Funzione LIRE, funzione EURO

La funzione LIRE converte un importo espresso in lire nella valuta euro. La funzione EURO compie il cambio inverso: da euro a lire.

La sintassi da usare è la seguente:
$[LIRE; <importo in lire>] $[EURO; <importo in euro>]

Esempio 70. Utilizzo funzione EURO
In questo esempio alla variabile PIPPO viene assegnato il valore 193627.

Codice 180. Funzione EURO.

```
1 #LETS PIPPO=$[EURO; 100]
```

4.12 Variabili di sistema per la grafica

Esiste un numero fisso di variabili all'interno di ***3CadEvolution*** che fanno diretto riferimento alle caratteristiche comuni a tutti i componenti, quali, ad esempio, l'altezza, la larghezza e la profondità. La sintassi per le variabili di sistema è data dal nome della variabile preceduto dal simbolo dollaro ($). Il contenuto di una variabile di sistema può essere forzato tramite il comando #LET ad un valore stabilito dall'utente. Seguono ora i nomi di tutte le variabili di sistema utilizzabili in ***3CadEvolution*** .

Variabili di sistema per la grafica

Variabile	Descrizione
$L	Lunghezza
$A	Altezza
$P	Profondità
$NL	Taglio in larghezza
$NA	Taglio in altezza
$NP	Taglio in profondità
$UL	Ulteriore lunghezza box
$UA	Ulteriore altezza box
$UP	Ulteriore profondità box
$PM	punto applicazione maniglia (1 sx, 2 centro, 3 dx)

Tabella 4.8: Variabili di sistema per la grafica.

Oltre a queste, il sistema mette a disposizione un altro gruppo di variabili, legate ai pulsanti presenti nella finestra di disegno della regola o di costruzione dell'ordine. Esse sono:

Variabili di sistema legati ai pulsanti

Variabile	Descrizione
$(LATOSX), $(LATODX)	Vengono generalmente usate per la gestione dello zoccolo laterale sinistro e destro nelle cucine, ma potrebbero avere impieghi diversi a discrezione dell'utente. L'operatore apponendo il flag nella relativa casella inizializza questa variabile a 1 e, quindi, all'interno della regola si può condizionare la presenza o meno di una tipologia in funzione del valore assunto dalla variabile in questione.
$(LATOASX), $(LATOADX)	Queste variabili hanno la stessa gestione delle precedenti, ma si riferiscono alle alzatine.
$(LATOASX), $(LATOADX), $(LATOBSX), $(LATOBDX)	Anche per queste variabili vale quanto spiegato in precedenza, solo che gestiscono le cornici dei pensili.
$(APERTO)	Si riferisce all'omonimo flag e viene usato per togliere i frontali dalla grafica. Anche in questo caso mettendo l'apice sulla casellina viene assegnato il valore 1 alla variabile APERTO che nella regola può essere verificata per condizionare la presenza o meno dei frontali.

Tabella 4.9: Variabili di sistema per la grafica legate ai pulsanti.

Facciamo notare al lettore che tutte queste variabili possono essere utilizzate anche nei cataloghi di interparete ad esempio per la gestione dei fianchi terminali.

Esempio 71. Scrittura di una regola
In questo esempio l'anta viene disegnata solo se $(aperto) ha valore zero, ossia il flag APERTO non viene settato.

Codice 181. Utilizzare le variabili di sistema legate ai pulsanti presenti nella finestra di disegno della regola.

```
#IF $(aperto)=0
   5 // anta
#ENDIF
```

Vediamo nella tabella 4.10 variabili di sistema che si riferiscono alle dimensioni e alle posizioni dei box.

Variabili di sistema utilizzati per i box

Variabile	Descrizione
CURBOX	Il numero del box corrente (la numerazione parte da zero)
CURREG	Il numero della regola se il box ne contiene più di una.
LBOX	Il numero del box collegato al box corrente (da notare che il conteggio in questo caso parte da 1).
LATO	Il numero del collegamento; ovvero il tipo di collegamento tra il box corrente ed il box indicato dalla variabile LBOX (vedremo in seguito la classificazione dei vari tipi di collegamento).
POSL	Posizione iniziale assoluta in larghezza del box.
POSA	Posizione iniziale assoluta in altezza del box.
POSP	Posizione iniziale assoluta in profondità del box.
SPTL	Spostamento in larghezza rispetto al box precedente e al tipo di collegamento in larghezza.
SPTA	Spostamento in altezza rispetto al box precedente e al tipo di collegamento in altezza.
SPTP	Spostamento in profondità rispetto al box precedente e al tipo di collegamento in profondità.
BAXY	Angolo attuale di rotazione sul piano XY del box corrente.
BAYZ	Angolo attuale di rotazione sul piano YZ del box corrente.
_TAGLIO	Questa variabile memorizza il taglio che l'utente imposta nella fase di progettazione, facendo clic destro su un oggetto selezionato.

Tabella 4.10 : continua nella prossima pagina

Tabella 4.10 : continua dalla pagina precedente

Variabile	Descrizione
_LINK	Questa variabile riporta il numero di addlink e tutti i parametri associati (prog descrizione, tipolink, sl, sa, sp, axz, axy, ayz), nel caso venga usato il comando !SCEGLIADDLINK (vedi spiegazione dei comandi).
_PARETE	Questa variabile 'sente' la vicinanza delle pareti (inserite con le opportune funzioni nella costruzione dell'ordine), assumendo il valore 1 oppure 3 rispettivamente se il muro è a sinistra o a destra della regola.
_ANGOLO	La variabile memorizza l'eventuale angolo fuori squadra delle pareti e consente alla base ad angolo di adattarsi ad esso.

Tabella 4.10: Variabili di sistema per la grafica legate ai box.

4.13 Variabili

In ***3CadEvolution*** esistono due tipi di variabili: quelle definite dall'utente, a valore singolo, e quelle che fanno riferimento agli elementi delle tabelle varianti.

Variabili definite dall'utente: L'utente che si occupa della stesura delle regole, a volte trova necessario poter disporre di variabili aggiuntive per memorizzare un valore, effettuare un calcolo od un confronto. A questo scopo ***3CadEvolution*** permette di creare semplici variabili che contengono un singolo valore; il loro nome deve essere preceduto dal simbolo $ e va racchiuso tra parentesi tonde, anche se quest'ultima condizione non è necessaria (ma preferibile per una maggiore leggibilità). Le variabili definite dall'utente non necessitano di dichiarazione; esse vengono create la prima volta che il valutatore incontra il loro nome durante l'interpretazione di una regola. Il valore di una variabile definita dall'utente viene assegnato mediante l'uso dei comandi #LET oppure #LETS, ma in questo caso non sarà preceduta dal simbolo $.

Riferimento agli elementi di una tabella varianti: Come abbiamo già spiegato nel corso del manuale, ***3CadEvolution*** permette di specificare, attraverso un'apposita finestra di inserimento, delle tabelle denominate varianti attraverso le quali definiamo le caratteristiche diverse dei vari componenti (colori, modelli, accessori etc.). Le informazioni all'interno di queste tabelle, sono organizzate per righe e colonne (similmente ad un foglio di calcolo). Per leggere uno dei valori di una tabella è necessario specificare, attraverso un'opportuna sintassi: il nome della tabella Variante

ed un indice di colonna che può essere numerico o alfabetico (non entrambi); l'indice assume valori scalari (ovvero ordinabili), che nel caso numerico sono i simboli dall'uno al nove (1-9), mentre nel caso alfabetico sono tutte le lettere dell'alfabeto inglese. La riga sarà invece quella attualmente selezionata durante l'utilizzo della regola. I valori di una tabella non possono essere alterati attraverso il comando di assegnazione #LET ma con il comando #LETV.

4.14 Attivazione delle tipologie

L'attivazione di una tipologia elementare avviene riportando all'inizio di una riga vuota nella fase della valutazione di una regola, il codice che identifica la tipologia stessa. Se non vengono specificati altri parametri, la tipologia viene disegnata nella sua definizione originale e cioè seguendo le caratteristiche definite all'interno della tabella delle tipologie.

Per apportare delle modifiche alle impostazioni di base, sarà necessario scrivere il nome di ciascun parametro seguito dal carattere '=' e dal nuovo valore che dovrà assumere. I vari parametri devono essere separati da [,].

Esempio 72. Attivazione di tipologia semplice e di tipologia con parametri.
Attivazione semplice : 3
Attivazione con parametri:

Codice 182. Apportare delle modifiche alle impostazioni di base.

```
3, C1=2, C2=2, L=$L-20, FAC=21
```

4.14.1 Parametri di attivazione delle tipologie

Parametri delle tipologie

Parametro	Descrizione
LATO	assume diversi significati in base al suo valore. Generalmente le tipologie grafiche lo hanno impostato a zero.
COD	oppure #C serve a specificare il codice che la tipologia genera all'interno della corrispondente riga del preventivo.
SL	serve a specificare uno spostamento in larghezza.
SA	serve a specificare uno spostamento in altezza.
SP	serve a specificare uno spostamento in profondità.

Tabella 4.11 : continua nella prossima pagina

Tabella 4.11 : continua dalla pagina precedente

Parametro	Descrizione
L	serve a specificare la dimensione in larghezza della tipologia.
A	serve a specificare la dimensione in altezza della tipologia.
P	serve a specificare la dimensione in profondità della tipologia.
C1, C2	si riferiscono rispettivamente al colore struttura e al colore aggiunto della tipologie. Generalmente solo il primo è importante, il secondo viene usato solo in presenza di particolari finiture anta (vedi figura sopra), dove è necessario distinguere il colore del pannello dall'eventuale telaio. I due parametri devono essere utilizzati sempre in coppia, anche se si vuole modificare solo uno dei due colori; si può fare riferimento direttamente ad un valore della tabella dei colori, oppure ad una variante tabellata.
FAC	serve per modificare il tipo di finitura della tipologia; il valore a cui si fa riferimento è uno dei codici delle finiture presenti nella scheda colori e finiture.
MAN	serve per modificare il tipo di finitura della tipologia; il valore a cui si fa riferimento è uno dei codici delle finiture presenti nella scheda colori e finiture.
AZ, AY, AX	serve per modificare l'angolo di rotazione con cui disegnare la tipologia; i tre valori numerici si riferiscono ai tre assi di posizionamento tridimensionale di un oggetto.
DIS	serve per assegnare una sagoma. Abbiamo già accennato alla possibilità di attribuire alla tipologia una grafica diversa dal parallelepipedo creando una cosiddetta Sagoma. Vedremo nelle prossime pagine i vari metodi per generare le sagome.
SE	condizione inline: imposta una formula, che se falsa non crea il pannello. In questo caso la tipologia 10 viene disegnata solo se la variabile ANTA vale uno.

Tabella 4.11 : continua nella prossima pagina

Tabella 4.11 : continua dalla pagina precedente

Parametro	Descrizione
X1, X2 ... X10	parametri aggiuntivi, definiti dall'utente e legati alla finitura con gli script e alle gestioni di distinta base e listini speciali.

Tabella 4.11: Parametri per le tipologie.

Esempio 73. Parametro SE

Codice 183. Parametro SE.

```
10, SE=$(ANTA)=1
```

4.14.2 Utilizzo e gestione delle macro di tipologie

Le macro non sono altro che regole richiamabili, a loro volta, all'interno di altre. Si accede alla creazione di una macro dal menù Regole-Tipologie e scegliendo la scheda macro.

Per attivare una macro all'interno di una regola, basta inserirne il nome all'inizio di una riga vuota. Come le tipologie, anche le macro ora possono essere modificate utilizzando dei parametri. Se, ad esempio, una macro disegna una scocca, la stessa potrà essere ridimensionata, ruotata o spostata in base ai diversi parametri a disposizione. I parametri standard di attivazione sono gli stessi spiegati per le tipologie.

Eventuali parametri aggiuntivi possono essere impiegati utilizzando all'interno della macro la funzione.

La sintassi da usare è la seguente:
!PARAMETRI $(nomevariabile1)=, $(nomevariabile2)=, ecc.

Quando viene richiamata la macro, si assegnano i valori ai parametri aggiuntivi con le stesse modalità di quelli standard.

Esempio 74. Macro CASSETTO

Codice 184. Assegnare parametri aggiuntivi.

```
!PARAMETRI CAS, COLCAS
14, SL=1, SA=$(SPOST), A=$(ALT) // cassetto
```

Esempio 75. Regola che disegna una cassettiera con quattro cassetti. Come viene richiamata e gestita la macro CASSETTO: Nella regola la variabile $SPALT viene assegnata per gestire lo spostamento in altezza dei cassetti.

```
1
2  CASSETTO,spost=$SA,alt=$(HCAS1:3)-3,CAS=$(cas1)
3  #LET SPALT=$SA+1+$(hcas1:3)+1
4
5  CASSETTO,spost=$(SPALT),alt=$(HCAS2:3)-3,CAS=$(cas2)
6  #LET SPALT=$SA+1+$(hcas1:3)+1+$(hcas2:3)+1
7
8  CASSETTO,spost=$(SPALT),alt=$(HCAS3:3)-3,CAS=$(cas3)
9
10 #LET SPALT=$SA+1+$(hcas1:3)+1+$(hcas2:3)+1+$(hcas3:3)+1
11
12 CASSETTO,spost=$(SPALT),alt=$(HCAS4:3)-3,CAS=$(cas4)
```

4.14.3 Macro Top

Questa gestione è tipica dei cataloghi di bagni, dove i piani hanno varie lavorazioni in base allo sviluppo della composizione e generalmente non hanno un conteggio a metro lineare (come avviene nelle cucine).

L'ambiente Macro Top è una delle schede che si trovano selezionando la voce Macro Top dal menù Regole. L'ambiente che si presenta è uguale a quello delle macro, visto prima. Anche in questa sede si deve andare a creare una regola, la quale, in base alle dimensioni e alle lavorazioni dei piani, riproduca i relativi codici esatti di vendita.

In relazione all'esigenza di avere le dimensioni totali dei piani, sono state introdotte le seguenti variabili:

- _L: riporta la lunghezza totale del top;
- _P: riporta la profondità totale del top.

Per quel che riguarda le lavorazioni, che nella regola vengono riportate nel codice con la sintassi COD=x/ /Lnnn , nell'ambiente del ricalcola top la variabile che le contiene si chiama _LAVTOP.

Esempio 76. Costruzione del codice in funzione della profondità del piano

Codice 185. Profondità del piano.

```
1 #IF $(_p) < 180+6
2    00BTA25P, $(top:@)
3 #ELSEIF $(_p) < 355+6
4    00BTB25P, $(top:@)
```

```
#ELSEIF $(_p < 480+6
   00BTD25P, $(top:@)
#ENDIF
```

Facciamo notare al lettore che il codice del piano viene scritto senza nessuna particolare assegnazione, sottolineiamo inoltre la presenza della variante dopo la virgola.

Come visto per la gestione delle macro di tipologie, anche per la Macro Top si vanno a definire in sostanza delle regole aventi un nome ed una descrizione. Il ricalcola top viene richiamato dalla relativa tipologia del piano nel campo commento, facendo precedere il nome dal carattere diesis o cancelletto (#).

5 Stampe parametriche

Le stampe di ***3CadEvolution*** sono gestite in modo parametrico. Sulla base dei dati prodotti per la stampa da programma, l'aspetto grafico del modulo, può essere completamente manipolato dall'operatore il quale può impostare la grafica della stampa a proprio piacimento. In alcuni casi, a seconda della stampa, l'operatore può anche intervenire sulla selezione dei dati in uscita.

I file dei moduli di stampa vengono creati e modificati utilizzando l'Editor *ASCII* di ***3CadEvolution*** o qualsiasi editor di preferenza dell'operatore; infatti un modulo di stampa è un file di tipo testo, simile a quello di memorizzazione delle regole ed avente l'estensione .STN .

I file delle regole di stampa sono contenuti nella cartella `_ecadpro /EcadPro/_EcadPro/Stampe`.

5.1 Gruppi della regola di stampa

La regola di stampa, nella modalità editabile, appare come una normale regola, Figura 5.1:

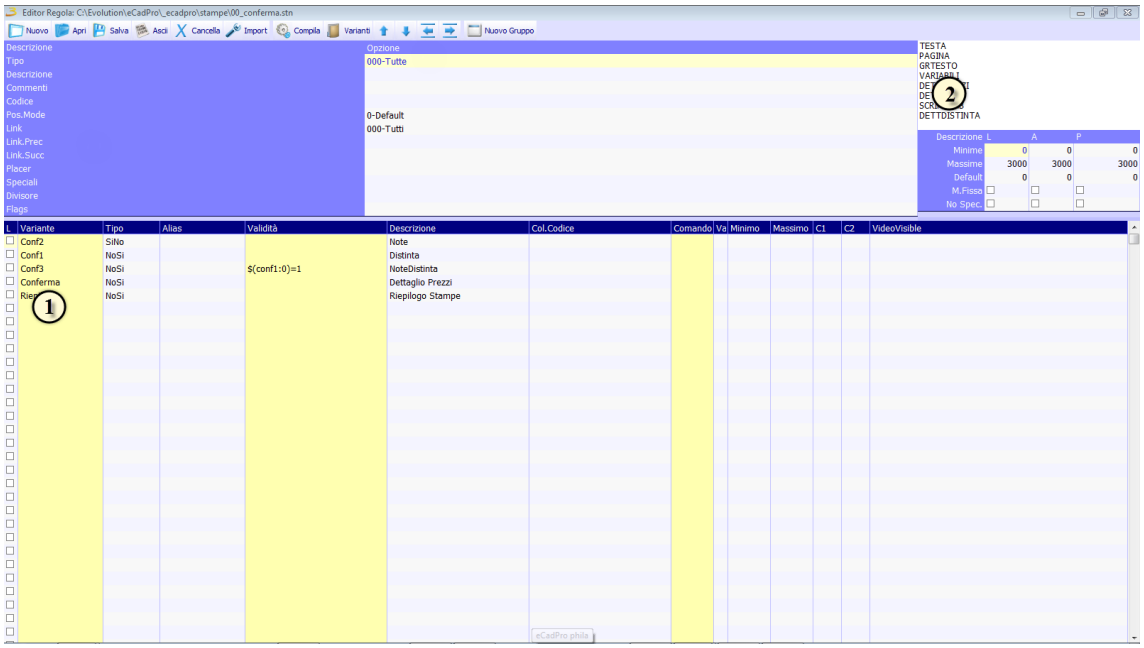


Figura 5.1: Regola di stampa.

(1) *Varianti della regola di stampa. Verranno richiamate all'interno della regola per porre dei controlli nella stampa di vario genere.*
(2) *Gruppi della regola di stampa.*

Sono presenti diversi gruppi personalizzabili nella regola di stampa, di cui i principali sono [TESTA] e [PAGINA]:

- [TESTA]: questo gruppo riguarda tutte le informazioni della testata dell'ordine(logo, numero dell'ordine, listino, data, informazioni sulla ditta e del cliente, totali, ecc.);
- [PAGINA]: questo gruppo definisce l'aspetto grafico del corpo di stampa. E' in questa parte del modulo che è contenuta la maggior parte dei dati di stampa.

È possibile poi aggiungere o rimuovere gruppi a seconda delle esigenze.

Il gruppo [PAGINA] serve in sostanza a creare le stampe multipagina, però è possibile definire tutta la stampa all'interno del gruppo [TESTA] includendo i comandi per creare le pagine:

!P : vai a pagina successiva, vuota;

!T : vai a pagina successiva, mantenendo l'intestazione.

5.2 Variabili della regola di stampa

I dati (le informazioni pre-impostate da programma o create tramite l'uso delle specifiche di comando), che vengono stampati in un modulo di stampa, possono essere contenuti in 4 tipi di variabili: variabile definita dall'utente, variabile predefinita, variabile di un record predefinito e variabile della regola `GENE.erg`. Un esempio è il nome di un cliente che varia da ordine a ordine, quindi, non può essere riportato nel file di stampa come un testo fisso definito dall'utente, ma deve essere riportato come una variabile di un record predefinito di ***3CadEvolution*** poiché assume un valore diverso di volta in volta. Da qui si comprende l'importanza dell'uso delle variabili nelle stampe parametriche.

Nelle regole di stampa(moduli) le variabili vengono indicate racchiudendo il loro nome tra parantesi angolari:

La sintassi da usare è la seguente:

<nomevariabile>

Codice 186. Variabile che riporta il numero di pagina attuale.

```
1 <PAG>
```

3CadEvolution gestisce quattro tipi di variabili:

- variabili definite dall'utente: il nome e il valore vengono gestiti dall'utente in maniera del tutto indipendente. Questa variabile è un valore che può essere richiamato molte volte nel corso del modulo di stampa, e che può inoltre essere modificato in qualsiasi momento. L'uso di una variabile rispetto all'uso diretto del valore risulta molto più flessibile e in alcuni casi indispensabile.

- variabili predefinite: sono variabili singole il cui nome e valore sono impostati da ***3CadEvolution*** in fase preliminare rispetto all'avvio della stampa, oppure impostate con l'uso delle specifiche di comando durante la stampa. Per usarle bisogna conoscerne il nome (una lista delle variabili generate da ***3CadEvolution*** viene fatta nelle prossime sezioni del capitolo);
- variabili dei record predefiniti: si tratta di un insieme di dati strutturati come un foglio elettronico o una tabella, i cui valori vengono inseriti in una specie di matrice di righe e colonne. Le colonne contengono i campi di un particolare record predefinito, mentre le righe contengono tutti i campi delle varie tipologie di record predefiniti da ***3CadEvolution*** da utilizzare in stampa. Più avanti vedremo al dettaglio l'elenco di ogni singolo record predefinito di ***3CadEvolution*** per la stampa in ambiente cucina o parete. Per utilizzare un valore (campo) memorizzato nei vari record, non occorre specificare il nome della variabile, ma è sufficiente indicare la riga (il numero del record) e la colonna (numero del campo relativo allo stesso record predefinito).
- variabili definite nella regola `GENE.ERG`: questo tipo di variabile viene dichiarato nel gruppo [GENE] della regola `GENE.ERG` che si trova nella cartella del catalogo. Per creare in aggiunta alcune variabili nella regola GENE.ERG è necessario utilizzare la finestra Editor Regole di ***3CadEvolution*** oppure usare un editor di testo generico. Le variabili dichiarate nella regola `GENE` differiscono da quelle dichiarate localmente in un modulo di stampa poiché si possono usare da qualsiasi regola o modulo di stampa.

Variabili predefinite

Durante l'utilizzo di ***3CadEvolution*** vengono predisposte da programma un insieme di variabili (predefinite) che si possono poi utilizzare all'interno delle regole di stampa e di cui non serve la dichiarazione. Queste variabili vengono elencate nella tabella 5.1 con la relativa spiegazione del contenuto.

In alcune stampe si possono incontrare delle variabili predefinite particolari di cui si riporta in seguito una lista a parte. Stiamo parlando delle stampe relative alla gestione dei carichi che costituiscono un insieme distinto nell'ambito delle regole di stampa di ***3CadEvolution*** (.STG) .

Esiste poi un ulteriore insieme di variabili predefinite (di cui si fornisce una terza lista) che si utilizzano in maniera diversa rispetto a tutte le altre variabili di stampa: sono le cosiddette variabili di programmazione; esse si usano all'interno delle regole di stampa in corrispondenza di particolari istruzioni di calcolo per effettuare una selezione dei dati da stampare. Il loro valore viene prodotto da programma.

Variabili predefinite standard per le stampe

Nome	Descrizione
DATA	data corrente
PAG	numero di pagina corrente
DIML	larghezza della composizione
DIMA	altezza della composizione
DIMP	profondità della composizione
RIGA	contatore di riga in una composizione
NOMEVARIANTE	restituisce il codice dell'opzione corrente selezionata per la variante
!NOMEVARIANTE	restituisce la descrizione dell'opzione corrente selezionata per la variante
DITTA0	logo
DITTA1	prima riga nome ditta
DITTA2	seconda riga nome ditta
DITTA3	indirizzo-località
DITTA5	fax
NOTA	note di stampa
RIGA	ultima riga ordine letta

Tabella 5.1: Variabili predefinite standard per le stampe.

Variabili predefinite gestione carichi per le stampe

Nome	Descrizione
CAR_COD	codice carico
CAR_DES	descrizione carico
CAR_VOLUME	volume del carico
CAR_COLLI	numero colli carico

Tabella 5.2: Variabili predefinite gestione carichi per le stampe.

Variabili di programmazione per le stampe

Nome	Descrizione
$(CURBOX)	Numero del box corrente

Tabella 5.3 : continua nella prossima pagina

Tabella 5.3 : continua dalla pagina precedente

Nome	Descrizione
$(SELECT)	clausola Select di una istruzione *SQL* di selezione
$(FROM)	clausola From di un'istruzione *SQL* di selezione
$(WHERE)	clausola Where di un'istruzione *SQL* di selezione
$(ORDER)	clausola Order di un'istruzione *SQL* di selezione
$(TABELLA)	codice tabella delle tabelle generiche di ***3CadEvolution***
$(DESTABELLA)	descrizione tabella delle tabelle generiche di ***3CadEvolution***

Tabella 5.3: Variabili di programmazione per le stampe.

5.3 Istruzioni per la formattazione della pagina

Le istruzioni di stampa sono un insieme di comandi particolari che servono per produrre in stampa linee, testi, tabelle di dati e comunque riprodurre su carta quanto prodotto a livello di calcolo. In questa sezione sono riportate le istruzioni riguardanti la formattazione della pagina.

ORIZZONTALE o VERTICALE

imposta l'orientamento della stampa in senso orizzontale o verticale.

PAPERSIZE

imposta le dimensioni del foglio di stampa: qui di seguito i parametri preconfigurati.

La sintassi da usare è la seguente:

```
PAPERSIZE param1
```

1 - Letter, 8 1/2 x 11 in.

2 - Legal, 8 1/2 x 14 in.

3 - Executive, 7 1/2 x 10 1/2 in.

4 - Tabloid, 11 x 17 in.

5 - Ledger, 17 x 11 in.

6 - Statement, 5 1/2 x 8 1/2 in.

7 - Folio, 8 1/2 x 13 in.

8 - A3, 297 x 420 mm

9 - A4, 210 x 297 mm

10 - A5, 148 x 210 mm

11 - B4, 250 x 354 mm

12 - B5, 182 x 257 mm

13 - Quarto, 215 x 275 mm

14 - Custom Size

DIM

imposta manualmente le dimensioni del modulo di stampa.

La sintassi da usare è la seguente:

```
DIM param1 param2
```

- param1: dimensione orizzontale;
- param2: dimensione verticale.

N.B: Le istruzioni ORIZZONTALE, VERTICALE, PAPERSIZE (o DIM), sono istruzioni obbligatorie, cioè devono essere sempre utilizzate nelle regole di stampa perché definiscono le caratteristiche di impostazione della pagina. E' importante, inoltre, fare attenzione alle dimensioni specificate con l'uso dell'istruzione PAPERSIZE (o DIM) poiché esse devono essere in armonia con l'orientamento di pagina impostato.

Esempio 1. Utilizzo dei comandi DIM e FONT nella stampa.

```
1 [TESTA]
2 VERTICALE
3 DIM 197 285
4 FONT TAHOMA
```

LI

traccia una linea secondo i parametri (è un'istruzione molto utilizzata poiché permette di definire le fiancature del modulo).

La sintassi da usare è la seguente:

LI param1 param2 param3

- param1: coordinate iniziali della linea; specificare le coordinate numeriche del punto iniziale della linea;

  (Xi , Yi)

- param2: coordinate finali della linea; specificare le coordinate del punto finale della linea;

  (Xf , Yf)

- param3: tipo di linea e spessore della linea; i due valori vanno digitati di seguito senza inserimento di spazi. Per il tipo di linea specificare uno dei seguenti valori:
  - N linea normale;
  - P linea punteggiata.

  Per lo spessore linea, specificare un numero (1= spessore minimo della linea).

Esempio 2. Utilizzo comando LI

Codice 187. Comando LI.

```
1 LI (100,100) (150,200) N2
```

TX

inserisce del testo secondo i parametri (anche questa istruzione è molto utilizzata perché serve per stampare testi fissi o il contenuto di vari tipi di variabili viste in precedenza).

La sintassi da usare è la seguente:

TX param1 param2 param3 param4

- param1: allineamento orizzontale e allineamento verticale: i due valori vanno digitati di seguito senza inserimento di spazi; Per l'allineamento orizzontale specificare uno dei seguenti valori:

- T (top): per allineare il testo in alto;
- B (buttom): per allineare il testo in basso.

Per l'allineamento verticale specificare uno dei seguenti valori:

- L (left): per allineare il testo a sinistra;
- C (centre): per allineare il testo al centro;
- R (right): per allineare il testo a destra.

- param2: dimensione del carattere e attributi, i due valori vanno digitati di seguito senza inserimento di spazi;
  Per la dimensione carattere specificare un numero.
  Per gli attributi (che sono facoltativi) specificare uno o più di uno dei seguenti valori:
  - C per formattare il testo in corsivo
  - S per formattare il testo in sottolineato
  - B per formattare il testo in grassetto
- param3: inizio del testo: specificare uno dei seguenti valori: (Xi , Yi) coordinate iniziali del testo X per specificare che il testo va inserito di seguito all'ultimo elemento inserito Y per specificare che il testo va inserito sotto l'ultimo elemento inserito
- param4: testo da stampare: può essere una stringa (scritta senza delimitatori), una variabile (scritta tra parentesi angolari) o un campo di un record (specificato con <riga,colonna>).

Esempio 3. Utilizzo comando TX

Codice 188. Comando TX.

```
1 TX TC 10BC (10,20) Cliente : stampo la dicitura Cliente
2 TX TL 12 (70,80) <DATA>
```

FONT

specifica il tipo di font da utilizzare da una certa riga del modulo di stampa fino alla specifica di un font diverso, in una successiva riga del modulo, o fino alla fine del modulo.

La sintassi da usare è la seguente:

FONT param1

param1: nome del font.Si specifica il nome di un font di *Windows* (vedi cartella `Fonts` del Pannello di Controllo). Se si specifica un font inesistente, ***3CadEvolution*** verrà terminato.

Esempio 4. Utilizzo comando FONT

Codice 189. Comando FONT.

```
FONT Times New Roman
```

WMF

riproduce un elemento grafico in una certa posizione del foglio.

La sintassi da usare è la seguente:

WMF param1 param2 param3

- param1: coordinate iniziali dell'elemento; specificare le coordinate dell'angolo sinistro in alto del grafico

  (Xi , Yi)

- param2: coordinate finali dell'elemento; specificare le coordinate dell'angolo destro in basso del grafico

  (Xd , Yd)

- param3: percorso file; specificare il nome del file relativo all'elemento grafico indicandone il percorso completo. Suggeriamo di utilizzare a tale proposito la variabile $(.percorso) che restituisce il percorso del catalogo in uso.

Esempio 5. Utilizzo comando WMF

Codice 190. Comando WMF.

```
WMF (7,1) (45,20) \$(.PERCORSO)\\LOGO.JPG
WMF(1,1) (55,21) \$(.percorso)\\stampe\\logo.wmf
```

CODE25

stampa il codice a barre di tipo INTERLEAVD 25 (il codice INTERLEAVD25 è di tipo numerico).

La sintassi da usare è la seguente:

CODE25 param1 param2 param3

- param1: coordinate iniziali dell'elemento; specificare le coordinate numeriche del punto iniziale del codice da stampare .

(Xi , Yi)

- param2: dimensioni del codice; specificare le dimensioni numeriche del codice a barre da stampare.

(Xd , Yd)

- param3: codice; specificare il codice da stampare.

Esempio 6. Utilizzo comando CODE25

Codice 191. Comando CODE25.

```
1 CODE25 (100,100) (50,20) 12345
```

CODE25V

vale quanto detto per il CODE25, solo che consente di stampare in verticale.

La sintassi da usare è la seguente:

CODE25V param1 param2 param3

CODE39 stampa il codice a barre di tipo 3D9 (il codice 3D9 è un codice di tipo alfanumerico).

La sintassi da usare è la seguente:

CODE39 param1 param2 param3

- param1 : coordinate iniziali dell'elemento; specificare le coordinate numeriche del punto iniziale del codice da stampare.

(Xi , Yi)

- param2: dimensioni del codice; specificare le dimensioni numeriche del codice a barre da stampare.

(Xd , Yd)

- param3: codice; specificare il codice da stampare.

Esempio 7. Utilizzo comando CODE39.

Codice 192. Comando CODE39.

```
1 CODE39 (100,100) (50,20) 3D9
```

CODE39V

oltre al CODE39 esiste anche la possibilità di stampare in verticale con CODE39V, i parametri sono gli stessi di CODE39.

La sintassi da usare è la seguente:

CODE39V param1 param2 param3

CODE39CK e CODE39CKV: si può stampare in CODE39CK e CODE39CKV cioè con il check digit(controllo di lettura).

CODEMABRY

per poter sfruttare le potenzialità di questo comando che consente di stampare tutti i tipi di codice a barra, l'utente deve essere in possesso dell'apposito programma.

La sintassi da usare è la seguente:

CODEMABRY param1 param2 param3 param4 param5 param6 param7

- param1: coordinate iniziali dell'elemento; specificare le coordinate numeriche del punto iniziale del codice da stampare.

  (Xi , Yi)

- param2: dimensioni del codice; specificare le dimensioni in larghezza e altezza del codice a barre da stampare.

  (DimX , DimY)

- param3: tipo, è il tipo di barcode (vedi tabella 5.4);
- param4: testo; è il testo da stampare;
- param5: direzione; è la direzione del barcode. E' opzionale ed il valore di default è zero.

  0 da SX a DX

1 da Dx a SX
2 da sopra a sotto
3 da sotto a sopra

- param6: allineamento; è l'allineamento del barcode all'interno del rettangolo di stampa. E' opzionale ed il valore di default è zero.

  0 sinistra
  1 destra
  2 centrato

- param7: dimensione barra; è la dimensione della singola barra. E' opzionale ed il valore di default è zero.

Tipo di barcode per le stampe

Valore	Funzionalità
0	niente
1	2 of 5 - numbers only
2	Interleaved 2 of 5 - numbers only
3	Code 3 of 9 (also called Code 39) - numbers, upper case, and - . * $ / + % (è il default)
4	Codabar (rationalized) - numbers and A B C D - $: / . +
5	Extended 3 of 9 (also called Extended Code 39) - full ASCII character set
6	Code 128A - upper case, numbers, punctuation, and control charac
7	Code 128B - upper and lower case, numbers, and punctuation
8	Code 128C - numbers only
9	UPC-A - numbers only (11 numbers and a check digit)
10	MSI Code - numbers only
11	Code 93 - numbers, upper case, and - . * $ / + %
12	Extended Code 93 - full ASCII character set
13	EAN-13 - numbers only (12 numbers and a check digit)
14	EAN-8 - numbers only (7 numbers and a check digit)
15	PostNet - numbers only (with a check digit) Note: this style always displays as full size. The BarWidth property has no effect
16	ANSI 3 of 9 (also called Code 39) - 3:1 narrow-to-wide bar width ratio - numbers, upper case, and - . * $ / + %
17	ANSI Extended 3 of 9 (also called Extended Code 39) - 3:1 narrow-to-wide bar width ratio - full ASCII character set

Tabella 5.4 : continua nella prossima pagina

Tabella 5.4 : continua dalla pagina precedente

Valore	Funzionalità
18	Code 128 (auto-select) - full ASCII character set - automatically selects between Code 128 A, B, and C to produce the smallest barcode
19	UCC/EAN-128 - special version of Code 128 used in package in HIBC applications - full ASCII character set
20	UPC-E - numbers only - used for zero-compression UPC symbols. For the Caption property you may enter either a six-digit UPC-E code or a complete 11-digit (includes code type, which must be 0 (zero)) UPC-A code. If an 11-digit code is entered, the BarCod control will convert it to a six-digit UPC-E code, if possible. If it is not possible to convert from the 11-digit code to the six-digit code, nothing is displayed
21	Royal Mail RM4SCC - numbers and letters only (with a check digit) This is the barcode used by the Royal Mail in the United Kingdom. Note: this style always displays as full size. The BarWidth property has no effect
22	MSI Code with 2 MOD checksum digits - numbers only
23	DUN-14 - numeric only. 14 digits long, including checksum digit. If you set the <Caption> to an 11-digit string, BarCod will modify it appropriately to create an EAN-13 code (to which it then adds the 14th digit, the checksum). If you set the <Caption> to a 12-digit string, BarCod will add the EAN-13 checksum and then the DUN-14 checksum. If you set the <Caption> to a 13-digit string or 14-digit string, BarCod will truncate the string to 13-digits and then add the DUN-14 checksum

Tabella 5.4: Tipo di barcode.

COLOREPENNA o PENCOLOR

cambia il colore della penna (cioè del bordo delle immagini o testi rappresentati in stampa). I parametri da specificare sono tre valori numerici delle intensità RGB del colore da settare e sono :

La sintassi da usare è la seguente:

COLOREPENNA param1 param2 param3

- param1: intensità del colore rosso; specificare il valore numerico corrispondente ;
- param2: intensità del colore verde; specificare il valore numerico corrispondente ;
- param3: intensità del colore blue; specificare il valore numerico corrispondente.

Esempio 8. Utilizzo comando COLORPENNA

Codice 193. Comando COLORPENNA.

```
1 COLOREPENNA 214 244 122
```

COLORERIEMPIMENTO o FILLCOLOR cambia il colore del riempimento di disegni o scritte. I parametri da specificare sono tre valori numerici delle intensità RGB del colore da settare e sono :

La sintassi da usare è la seguente:

COLORERIEMPIMENTO param1 param2 param3

- param1: intensità del colore rosso; specificare il valore numerico corrispondente ;
- param2: intensità del colore verde; specificare il valore numerico corrispondente ;
- param3: intensità del colore blue; specificare il valore numerico corrispondente.

Esempio 9. Utilizzo comando COLORERIEMPIMENTO

Codice 194. Comando COLORERIEMPIMENTO.

```
COLORERIEMPIMENTO 214 244 122
```

TIPORIEMPIMENTO o FILLSTYLE

cambia il tipo di riempimento di oggetti di tipo BX. Il parametro da specificare assume uno dei seguenti valori numerici.

La sintassi da usare è la seguente:

TIPORIEMPIMENTO param1

param1: tipo;

0 Riempimento pieno
1 Nessun Riempimento
2 Linee Orizzontali
3 Linee Verticali
4 Linee diagonali da destra a sinistra
5 Linee diagonali da sinistra a destra
6 Rete
7 Rete diagonale

Esempio 10. Utilizzo comando TIPORIEMPIMENTO

Codice 195. Comando TIPORIEMPIMENTO.

```
TIPORIEMPIMENTO 6
```

SPESSORE

cambia lo spessore della penna in stampa. Il parametro da specificare è il valore numerico dello spessore.

La sintassi da usare è la seguente:

SPESSORE param1

Esempio 11. Utilizzo comando SPESSORE

Codice 196. Comando SPESSORE.

```
SPESSORE 2
```

BW o MODOBW

viene utilizzata per settare la modalità di stampa in bianco e nero. Per default, infatti, la stampa è a colori.

ORIGINE

serve per modificare l'origine del blocco di stampa in una posizione diversa da quella standard che corrisponde all'angolo in alto a sinistra del foglio (0,0). Si usa ad esempio per la carta intestata.

La sintassi da usare è la seguente:

ORIGINE param1

param1: coordinate iniziali della stampa; specificare le coordinate numeriche del punto iniziale del blocco di stampa.

(Xi , Yi)

Esempio 12. Utilizzo comando ORIGINE

Codice 197. Comando ORIGINE.

```
ORIGINE (0,150)
```

5.4 Parametri per i comandi *ARG* e *TOP*

ARG

stampa un disegno di ***3CadEvolution*** .

La sintassi da usare è la seguente:

ARG param1 param2 param3 param4

- param1: coordinate iniziali del disegno; specificare le coordinate dell'angolo sinistro in alto del disegno.

(Xi , Yi)

- param2: coordinate finali del disegno; specificare le coordinate dell'angolo destro in basso del disegno.

(Xf , Yf)

- param3: vista del disegno ; specificare la vista con cui riprodurre il disegno. Essa assume i seguenti valori:

  V(angoloXY, angoloXZ, AngoloYZ) della vista

  oppure

  VNumeroVista, dove NumeroVista può assumere i seguenti valori:

  0 vista libera (quella attiva al momento della stampa)
  1 fronte
  2 profilo
  3 pianta
  4 ortogonale dx
  5 ortogonale sx
  6 ass. 45 dx dall'alto
  7 ass. 45 sx dall'alto

  E' possibile stampare anche le viste personalizzate salvate per l'ordine (v. manuale utente), per far questo il parametro diventa VSn, dove con n si precisa il numero della vista.

- param4: opzioni ; è un insieme di opzioni che permettono di avere un disegno con caratteristiche particolari.

Esempio 13. Utilizzo istruzione ARG.

Codice 198. Istruzione ARG.

```
1 ARG (1,1) (100,70) V1 /DM(600,3000,625) /BOX
2 ARG (50,25) (280,200) V0 /PA
3 ARG (5,35) (25,110) V4 /DM(600,3000,625) /F6 /R-2,0,0,A,B,BR
4 ARG (45,35) (205,110) V3 /DM(5000,3000,800) /F6 /R0,0,0,L
    ,1000,B,BR /R-2,0,0,A,B,BR
5 ARG (45,120) (205,170) V5 /DM(5000,3000,2000) /F6 /R0,0,0,P,B
    ,BR /TE*15*
```

TOP

stampa il disegno del top della composizione secondo i parametri specificati.

La sintassi da usare è la seguente:

TOP param1 param2 param3

- param1: coordinate iniziali del top; specificare le coordinate numeriche del punto iniziale del disegno del top

  (Xi , Yi)

- param2: coordinate finali del top; specificare le coordinate numeriche del punto finale del disegno del top

  (Xf , Yf)

- param3: opzioni ; è un insieme di opzioni che permette di avere un disegno con caratteristiche particolari.

In questa sezione sono presenti parametri, opzioni che possono essere utilizzati sia nel comando *ARG* sia *TOP*.

5.4.1 Parametro F

```
/F[N]
```

Definisce la dimensione del font. N può assumere i valori tra 0.1 e 10. Il valore di default è 1.

5.4.2 Parametro TI

```
/TI*NumTipologia*...*
```

Il disegno viene fatto usando solo le tipologie specificate come parametri. Il numero della tipologia da includere deve essere racchiuso tra asterischi.

5.4.3 Parametro TE

```
/TE*NumTipologia*...*
```

Il disegno viene fatto senza le tipologie specificate come parametri. Il numero della tipologia da escludere deve essere racchiuso tra asterischi.

Esempio 14. Stampa del disegno in vista ortogonale escludendo alcune tipologie.

```
1 ARG (0,0) (50,50) V4 /BR /TE*446*
```

5.4.4 Parametro PA

```
/PA
```

Il disegno viene fatto con l'ausilio delle pareti.

5.4.5 Parametro BI

/BI

Forza disegno in bianco e nero però con luminosità.

5.4.6 Parametro BR

/BR

Per stampare il disegno ignorando le tonalità di grigio, cioè stampa linee nere su sfondo bianco.

5.4.7 Parametro CO

/CO

Forza la stampa a colori.

5.4.8 Parametro TX

/TX[n]

Il parametro n può assumere due valori:

0 : per stampare senza materiali;

1 : per stampare con materiali.

5.4.9 Parametro LS

/LSlayer1*layer2*...*layerN

Nella grafica è possibile inserire simboli che sono dei oggetti bidimensionali ma posizionati nello spazio(prese, impianti, ecc.). Durante la sua definizione, che avviene tramite l'apposita funzione, al simbolo viene associato un layer. In fase di stampa, per default, i simboli non vengono visualizzati ma se lo si desidera, il comando */LS* visualizza nella stampa i simboli appartenenti ai layer specificati come parametri.

Con:

/LS-1

vengono attivati tutti layer.

5.4.10 Parametro BXI

```
/BXI*nBox*...*
```

Stampa box inclusi specificati dai parametri. Il numero del box da includere deve essere racchiuso tra asterischi ed è il campo nBox su rOrdine.

5.4.11 Parametro BXE

```
/BXE*nBox*...*
```

Stampa box esclusi specificati dai parametri. Il numero del box da escludere deve essere racchiuso tra asterischi ed è il campo nBox su rOrdine.

5.4.12 Parametro BXT

```
/BXT[T]
```

Stampa solo i box esclusi che contengono la tipologia passata come parametro.

5.4.13 Parametro SL

```
/SL[x]
```

E' lo spessore della linea, evidenzia il tratto a penna. Il valore di default è 0.

5.4.14 Parametro BW

```
/BW
```

Stampa in bianco e nero con le ombre.

5.4.15 Parametro EX

```
/EX[valore]
```

stampa l'esplosione del disegno, è necessario specificare il grado di esplosione desiderato, valore compreso tra 0 e 1.

5.4.16 Parametro Z

```
/Z[N]
```

Esegue uno zoom del disegno. Se vale 1 il disegno sarà rappresentato come a video, se incrementato, subirà lo zoom selezionato. Accetta anche valori con la virgola. Il valore N rappresenta il fattore zoom.

5.4.17 Parametro SCRIPT

/SCRIPT file

Questo comando esegue lo script contenuto all'interno del file passato come parametro. Ciò permette di fare delle stampe sofisticate utilizzando i comandi per le finiture e altri utilizzabili all'interno degli script.

5.4.18 Parametro GA

/GAx,x,x

Nelle stampe gestisce la percentuale gamma di: Ambiente, Diffusa e Speculare che già potevano essere modificati in Viste da Configuratore. Questi parametri servono in sostanza a regolare il grado chiaro/scuro delle stampe. Dove gli x sono numeri da 0 a 1 che distinguono le percentuali dei fattori di correzione. Per un utilizzo normale è meglio impostare tutti e 3 i valori con il medesimo valore. Lo 0 darà un disegno nero mentre l'1 darà un disegno di solo linee di costruzione e colori.

Esempio 15. Utilizzo parametro del /GA di ARG.
Nella Figura 5.5 sono riportati 3 esempi con il 20%, il 70% e 100% dei parametri gamma:

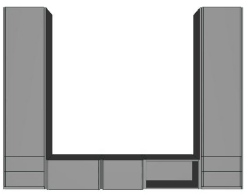


Figura 5.2: Il 20% del parametro gamma.

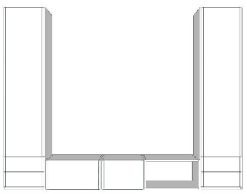


Figura 5.3: Il 70% del parametro gamma.

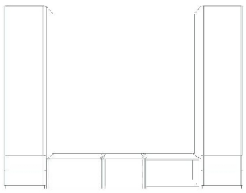


Figura 5.4: Il 100% del parametro gamma.

Figura 5.5: Immagini rappresentanti il chiaro/scuro delle stampe.

5.4.19 Parametro CA

/CA[x]

Correzione luce ambiente da 0 a 10.

5.4.20 Parametro BXTOPFRONT

/BXTOPFRONT[nBox]:tipPianta:tipFronte

permette per ogni comando *ARG* o *TOP* di stampare un singolo BOX(con placer collegati) visualizzando il box in questione contemporaneamente secondo una vista frontale ed una vista pianta, allineate l'una sotto l'altra.

Il comando /BOXTOPFRONT è formato da quattro parametri separati da [:] .

- nBox: identificativo del box da stampare(nBox su rOrdine).
- tipologie che vanno escluse od incluse nella vista pianta, vengono inserite tra [*];
- tipologie che vanno escluse od incluse nella vista fronte, vengono inserite tra [*];
- distanza in mm reali tra la vista fronte e la vista pianta.

Sia nella vista fronte che nella vista pianta è possibile scegliere se la lista delle tipologie va visualizzata o meno:

- specificare TE davanti alla lista delle tipologie per escluderle;
- specificare TI davanti alla lista delle tipologie per includerle;

Esempio 16. Regola di stampa con /BXTOPFRONT.
In questo esempio la variabile $(alfa) contiene il box da stampare. Nella stampa in pianta vengono escluse le tipologie 216 e 215. Invece alla stampa in fronte la tipologia 216 è aggiunta.

Codice 199. Parametro /BXTOPFRONT dell'istruzione ARG.

```
1  Verticale
2  DIM 200 280
3  #FORDBS alfa=select nbox from ecadmaster.dbo.rOrdine where
4  numero=$(testa_numero) and padre = 0 and nbox > 0
5  #LET idB = $(i)
6  #LETS idU = $(.idbox.4)
7     ARG (0,0) (200,233) V0 /SL1 /BI /CA1, 0 ,0
8     /BXTOPFRONT$(alfa) :TE216*215:TI261:500
9     !P
10 #ENDDBS
```

5.5 Parametri del comando *ARG*

5.5.1 Parametro ORDINA

/ORDINA

Numerazione dei box tramite pallini. Il programma è dotato di una funzionalità automatica di pallinatura che consente, in fase di stampa di numerare i box tramite dei pallini. Essa viene attivate in varie situazioni, una di queste è ad esempio il comando: *ARG /ORDINA*

Il programma posiziona in automatico i pallini al centro del box (il baricentro per essere più precisi). E' capitato in talune situazioni che tale posizionamento non sia corretto (es sovrapposizione con maniglie o con altre etichette).

E' pertanto prevista la possibilità di cambiare la posizione del pallino che compare. Per farlo è sufficiente inserire all'interno della regola del box interessato il comando:
!ADDPUNTO x,y,z,0,1

Questo comando definisce la posizione (x,y,z) in coordinate relative al box sulla quale saranno posizionati i pallini in stampa con il comando ARG /ORDINA.
Considerazioni:

- E' possibile inserire al massimo un segnaposto; se non è specificato alcun segnaposto il pallino viene posizionato nel baricentro dell'oggetto
- le coordinate x,y,z sono riferite all'origine del box;
- funziona anche con i figli di placer.

5.5.2 Parametro PR

```
/PR
```

Effettua stampa in prospettiva.

5.5.3 Parametro AS

```
/AS
```

Effettua stampa in assonometria.

5.5.4 Parametro BOX

```
BOX[N]
```

Stampa il box indicato dal parametro N(campo nbox su rOrdine).

5.5.5 Parametro SCALA

```
/SCALA
```

Applica il fattore di scala all'immagine.

> ! Viene applicato solo in assonometria e con impostazione per stampe > A4

5.5.6 Parametro QUOTE

/QUOTE*L1:D1*...*Ln:Dn*

Stampa le quote appartenenti all'elenco dei layer. I layer di quotatura sono quindici e vengono specificati nella lista dei parametri separati da * . Oltre ai layer vengono specificate le distanze delle quotature dal disegno. Le quote che verranno visualizzate sono quelle definite nella regola con il comando *!QUOTE*.

5.5.7 Parametro VC

/VC

All'interno del comando *ARG* è possibile stampare la vista Custom contenente le quote auto generate e quelle digitate a mano dall'utente tramite la funzione `.CUSTOM00`. La sintassi /VC dice al programma che si vuole stampare la vista *.CUSTOM00* ovvero si vuole stampare le quote automatiche più quelle messe manualmente dall'utente tramite l'apposito pulsante nella grafica.

> N.B. Le quote manuali sono messe nel layer 0 in automatico dal programma.

Esempio 17. Stampa delle quote globali.
Viene stampata la vista custom con tutte le sue quote ma vista in assonometria sinistra di -30°(/ANGLE4)

```
1 ARG (0,0) (280,185) /VC /ANGLE4
```

5.5.8 Parametro ANGLE

/ANGLE[x] o /ANGLE[angoloX], angoloY, fov

Permette di modificare il punto di vista della vista custom(utilizzato insieme al /VC). Il parametro x specifica il tipo di vista:

1 : frontale assonometrico;

2 : profilo assonometrico;

3 : pianta assonometrica;

4 : assonometria sinistra -30°;

5 : assonometria sinistra 30°;

6 : assonometria sinistra -45°;

7 :assonometria sinistra 45°.

Con la sintassi /ANGLE[angoloX], angoloY, fov è possibile personalizzare la vista. Dove:

angoloX: angolo dell'inquadratura sull'asse x;

angoloY: angolo dell'inquadratura sull'asse y;

fov: l'apertura focale della telecamera. Il valore -1 rappresenta assonometria, 60 l'apertura normale del programma.

5.5.9 Parametro ETI

```
/ETI
```

Stampa in trasparenza.

5.6 Parametri del comando *TOP*

5.6.1 Parametro QC

```
/QC[X]
```

Indica il tipo di quota che deve essere stampata: 0 il pallino, 1,2,3,4 rispettivamente le misure effettiva, reale, davanti e dietro della cornice. Se X=5 non viene stampata alcuna quota.

5.6.2 Parametro QD

```
/QD[X]
```

Imposta la distanza della quota dalla cornice, valore intero che può essere positivo o negativo per allontanare o avvicinare la quota. Il valore di default è 20.

5.6.3 Parametro M

```
/M[X]
```

Stampa il top con la matrice specificata.

5.6.4 Parametro T

/T[X]

Stampa il top con il numero specificato.

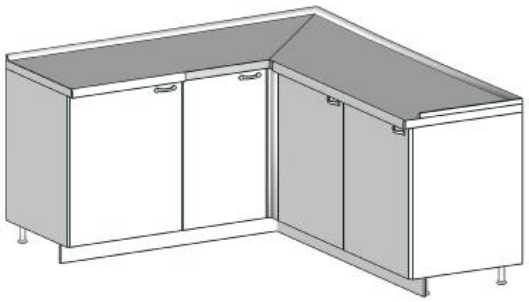
Esempio 18. Utilizzo istruzione TOP.

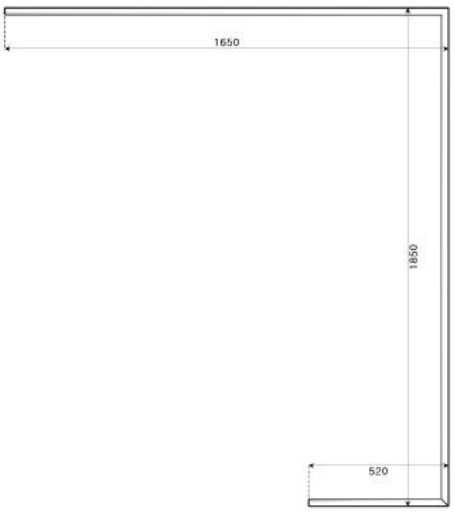
Codice 200. Istruzione TOP.

```
1   TOP (1,40) (199,250) /T4
2   TOP2 (121,25) (279,199) /M<AA> /SL2 /BR /QC1 /QD-100 con
       la cornice
```

cad evolution	*Riferimento*	**Alzatine**	*Data Ord.* **09/06/2009**
		Cliente **-**	*Ordine* **180025**

Codice	Descrizione	Dim.
SALZ01	Alzatina ALZ. LAMINATO 25X25 - 25x25 Nr.1 Lung.1650 (1650) Taglio Fin.-45° Nr.1 Lung.1850 (1850) Taglio Ini. 45° Taglio Fin.-45° Nr.1 Lung.520 (520) Taglio Ini. 45°	4020x25





5.7 Comando *NTAB*

NTAB

stampa una tabella con dei dati preimpostati. Questo comando funziona con chiamate *SQL*.

La sintassi da usare è la seguente:

NTAB(<param1>, <param2>, <param3>, <param4>) ([param5], [param6], [param7] , <param8>) (param9) (param10) ...

- param1 x1, y1: origine sinistra della tabella;
- param2 x2, y2: dimensioni x e y;
- param3 y1a: origine della seconda pagina se è diversa;
- param4 linee: indica se ogni riga va interrotta con una linea di separazione;
- param5: scriptgruppovbs: eventuale gruppo all'interno della stampa, dove processare ogni singola riga;
- param6, param7, ...: sottotab1, sottotab2,...: sottoquery per ogni riga, vedi per esempio *00_conferma.snt* o *distinta.stn*. Esiste una seconda *Query* eseguita per ogni riga, nel gruppo <sottoquery>;
- param8 = sql: comando *SQL* per la tabella: nTab lavora solo su chiamate *SQL*, ogni campo di ritorno diventa una variabile ed è processata prima della stampa da script `scriptgruppovbs`;
- param9, param10, ...: (descampo1, allineamento, nomecampo), (descampo2, allineamento, nomecampo),...

Esempio 19. Stampa di una tabella con /NTBA

```
1   NTAB (0,133,200,265,24,1) \
2 (scripttab,\$(DETTAGLIOPREZZI)) \ (selectcodart,des,xnote,qta
    ,pz,pr1,pr2,diml,dima,dimp,tipofm,\
3 sc1,sc2,sc3,sc4,var,riga from ecadmaster.dbo.rOrdine \
4 where percorso='\$(ditta)' and numero=<testa_numero> \
5 and (tiporiga='1' or tiporiga='3' or tiporiga='4') order by
    riga) \ // COMANDO SQL
6       (Codice , 15L ,, ^<CODART>^) \
7       (Descrizione, 40L ,, ^<des>^,<xnote>,<varianti>) \
8       (Qta , 6.2R ,, <qta>*<pz>) \
9       (sconti , 15L ,,[sconti;<sc1>;<sc2>;<sc3>;<sc4>]) \
10      (Totale , 8.2R \ ,,[Sconto;<pr1>+<pr2>;<sc1>;<sc2>;<sc3
          >;<sc4>]*<pz> ) \
11      (dimensioni, 12R,,^[dim;<diml>;<dima>;<dimp>]^ )
```

Esempio 20. Stampa.

Codice 201. Stampa.

```

VERTICALE // orientamento della pagina
PAPERSIZE 9 // VALORI PER PAPERSIZE (9 = A4, 210 x 297 mm)
WMF (0,0) (56,24) $(.percorso)\logo.wmf // inserimento
    immagine
FONT TAHOMA // font di scrittura
TX TL 16BC (59,1) $(TIPO) // inserimento variante
TX TR 16B (145,1) N.<testa_numero> // richiamo di variante di
     testata
TX TL 7BC (59,12) Riferimento cliente // inserimento testo

// inserimento di un riquadro
LI (192,246) (192,272) N1 // LINEA CHIUSURA DX
LI (0,246) (192,246) N1 // LINEA SUPERIORE
LI (0,246) (0,272) N1 // LINEA CHIUSURA SX
LI (0,272) (192,272) N1 // LINEA INFERIORE
LI (0,259) (192,259) N1 // LINEA DIVISIONE CENTRALE

// query per la distinta base
(select t.codcomp,a.des,sum(t.qtacomp) qta,t.diml,t.dima,t.
    dimp,t.n1,t.n2,t.n3,t.n4,t.fm \
       from ecadmaster.dbo.distintat t \
       left join articoli a on t.codcomp=a.cod \
       where t.percorso='$(.percorso)' and t.numero =<
           testa_numero> and t.riga=<riga> \
       group by t.codcomp,a.des,t.diml,t.dima,t.dimp,t.n1,t.n2
           ,t.n3,t.n4,t.fm order by t.codcomp)

(Codice , 15L ,, <codcomp>) \
       (Descrizione, 40L ,, <des>,<n1>,<n2>,<n3>,<n4>) \
       (Qta ,6 ,, <qta>) \
       (p.Unit , 6.2R ,,) \
       (sconti , 15L ,,) \
       (Totale , 8.2R ,,<fuorim> ) \
       (dimensioni, 12C,,\fs15 \ql <evidenzia> [dim;<diml>;<
           dima>;<dimp>;<FM>] )
```

5.8 Stampa per parete

Il comando ARG predispone di una modalità di stampa per parete.

/WALL

stampa la i-esima parete su cui sono appoggiati degli oggetti.

La sintassi da usare è la seguente:

/WALL$(i)

All'interno della stampa stessa è possibile definire dei layer di quotatura attraverso il flag /QW del comando ARG stesso:

Esempio 21. Stampa per parete.

Codice 202. Commando WALL.

```
1 ARG (0,0) (200,175) V0 /WALL$(1) /QW[...parametri_quotatura
    ...]
```

> ! Per poter utilizzare il parametro /QW deve esserci necessariamente /WALL.

/QW

La sintassi da usare è la seguente:

/QWparametri_quotatura

parametri_quotatura: è una stringa che contiene i vari layer di quotatura separati da *.

I layer sono composti da tre parametri che vengono separati tra loro da `:`.

layer:flag:descrizione

I parametri assumono il loro significato, tabella 5.5 in base al tipo di quotatura, che viene definita con il parametro layer.

Tipi di quotatura per stampa pareti

Tipo layer	Descrizione del layer	Significato Flag	Descr
0	Layer di quotatura verticale, quota tutti i box inseriti sulla parete facendo comparire a sinistra della stampa la quota in altezza	Tralasciare per questo tipo il flag	Tralasciare il parametro descrizione
1	Layer di quotatura orizzontale dei punti di riferimento della parete (PORTE, FINESTRE, ADDPUNTO,..)	Tralasciare il parametro flag mettendolo a 0	Specificare

Tabella 5.5 : continua nella prossima pagina

Tabella 5.5 : continua dalla pagina precedente

Tipo layer	Descrizione del layer	Significato flag	Descr
2	Layer di quotatura orizzontale dei box.	Flag contiene misura in altezza. Tutti gli oggetti che si trovano a quell'altezza verrano quotati da questo layer.	Specificate la descrizione.
3	Layer quotatura verticale degli scarichi.	Specificare come parametro flag il layer degli scarichi che si desiderano quotare.	Necessita di descrizione
4	Layer quotatura orizzontale degli scarichi.	Specificare come parametro flag il layer degli scarichi che si desiderano quotare.	Ignorata la descrizione.

Tabella 5.5: Tipi di quotature per la stampa pareti.

Esempio 22. Layer di quotatura per la stampa pareti (1).

```
/QW0*2:200:Basi*2:1700:Pensili
```

Dove:

0 - il layer di tipo 0

2:200:Basi - layer di tipo due con altezza di quotatura uguale a 200 e descrizione la stringa Basi.

2:1700:Pensili - layer di tipo 2 con altezza uguale a 1700

Come si può notare dall'esempio il layer con la descrizione Base ha l'altezza della quotatura uguale a 200 che tracciata in orizzontale incontra gli articoli Base per la quota, invece articoli Pensili si trovano all'altezza 1700.

Esempio 23. Layer di quotatura per la stampa pareti (2).

```
quotewalls=/QW0*2:200:Basi*2:1700:Pensili*1::Ingombri*4:2:
    Impianti*2:-1:Totale
ARG (0,0) (200,175) V0 /WALL$(i) $(quotewalls)
```

5.9 Stampa con righelli(Rullers)

E' possibile attivare sulla stampa tramite il comando *ARG/RULER* un righello che quoti dei punti particolari inseriti all'interno dei Box.

I punti di quotatura vanno inseriti all'interno dei box tramite una tipologia che abbia lato 42.

Aggiungere punti di quotatura:

All'interno dei box che volete quotare aggiungete una tipologia di lato 42, ad esempio:

Codice 203. Aggiungere delle tipologie di lato per visualizzare i rullers.

```
001,LATO=42,SL=0,SA=0,SP=0,x2=-1
001,LATO=42,SL=0,SA=$A,SP=0,x2=1
```

Il lato 42 è una tipologia che definisce il punto dello spazio di coordinate SL, SA, SP rispetto all'origine del box. Nell'esempio sopra indicato si è aggiunto un punto di quotatura nell'origine del box ed un altro in corrispondenza della sua altezza. Il parametro X2 serve per identificare su che layer andrà il punto di quotatura aggiunto.

Attivare i righelli in stampa:

In fase di stampa potete attivare i righelli tramite il comando */RULER*. Il programma andrà a quotare tutti i punti aggiunti tramite lato 42 tenendo in considerazione i seguenti parametri:

La sintassi da usare è la seguente:

ARG/RULER orienta, tipo, layers, maxMis, lunghLineeCostruzione, d0, d1, d2

- Orienta: definisce l'orientamento del righello e può assumere i valori:
  - 0 : righello verticale a sinistra del box più a sinistra inquadrato;
  - 1 : righello verticale a destra del box più a destra inquadrato;
  - 2 : righello orizzontale in alto;
  - 3 : righello orizzontale in basso.
- tipo: definisce lo stile grafico del righello:
  - 0 : righello con trattini stile metro;
  - 1 : quote tradizionali.
- layers: definisce quali layer quotare, sono separati da * e potete mettere -1 per attivare tutti i layer;

- maxMis: definisce la lunghezza effettiva del righello:
- lunghLineeCostruzione: le linee di costruzione sono delle linee sottili che vengono tracciate a partire dal righello in direzione degli elementi quotati. Servono per tracciare una sorta di griglia davanti al disegno:
  - -1 : allora le linee di costruzione vengono tracciate in automatico sulla dimensione massima del disegno;
  - 0 : nessuna linea di costruzione;
  - un valore positivo in millimetri: indica la lunghezza effettiva della linea di costruzione.
- D0: indica quanto staccato dal disegno si vuole il righello;
- D1: indica quanto staccato dal righello si vuole il primo layer di quotatura;
- D2: è la distanza tra i layer di quotatura.

5.10 Stampa con marker

I *marker* sono delle tipologie particolari che non sono rappresentate in grafica ma che portano al loro interno un CODICE. In fase di stampa specificando sull'ARG il parametro /MARKER il programma effettua due cose:

1. Numera con dei pallini i MARKER mettendo lo stesso numero su tutti i marker che hanno lo stesso codice.
2. Popola una variante al cui interno è contenuta la lista di codici presenti nella pagina stampata.

Generalità:

WrMarker è un particolare tipo di pannello (inseribile tramite finitura) le cui caratteristiche sono:

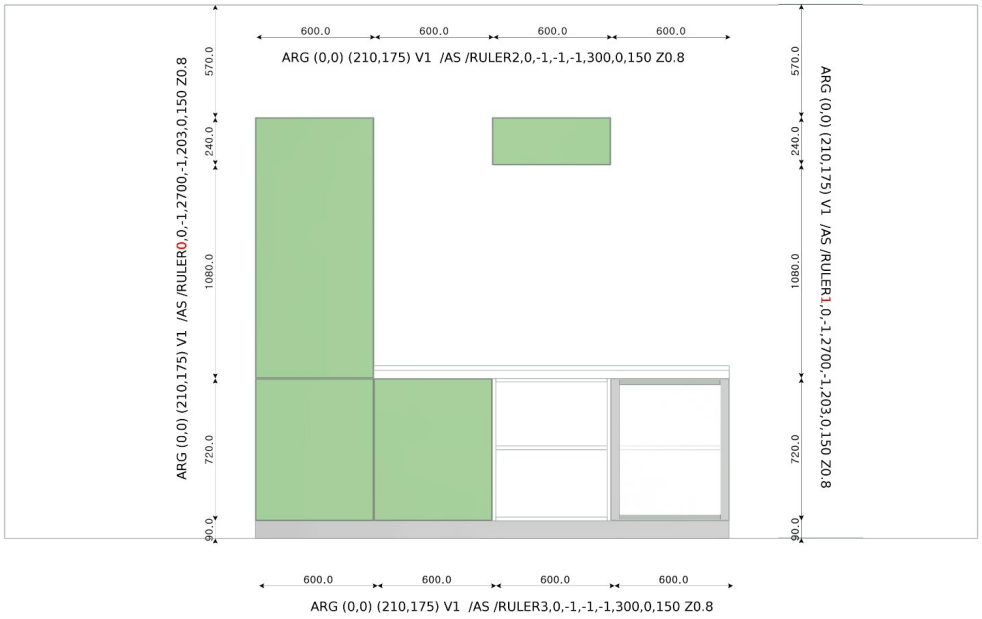
- E' invisibile in grafica, ma ha comunque dimensioni e posizione effettiva nello spazio;
- porta all'interno una descrizione;
- in stampa una volta specificato all'interno del comando *ARG* il parametro /MARKER si ottiene una pallinatura dei marker. Tutti i marker con la stessa descrizione vengono pallinati con lo stesso numero.

Da regola/finitura:

Inserire una tipologia con una finitura che chiama il comando *wrMarker*.

La sintassi da usare è la seguente:

```
wrMarker dl, da, dp, posizionePallino, stringaCodice, layer
```



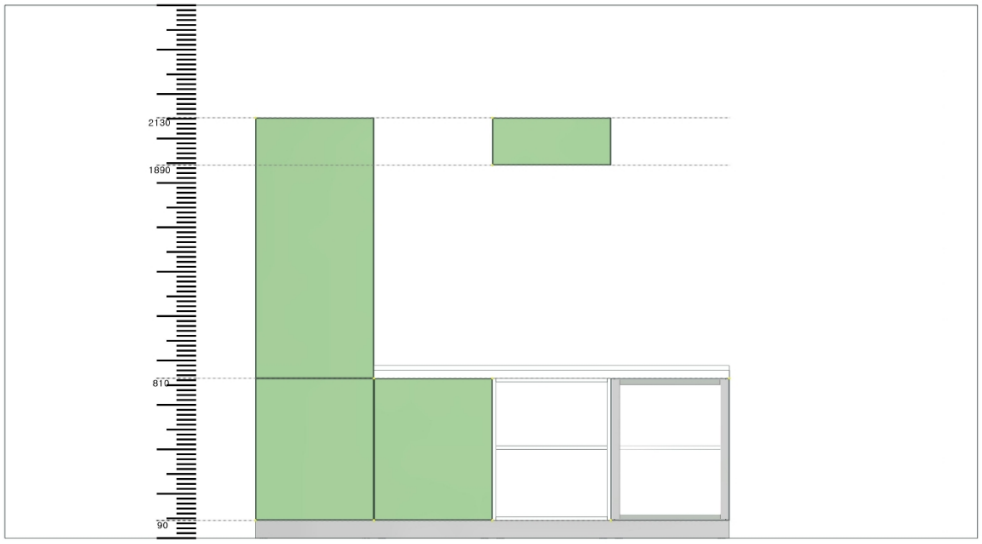


Figura 5.6: Stampe con ruller.

Dove:

- dl, da, dp: sono l'ingombro fisico su cui sarà posizionato il marker;
- posizionePallino: è un parametro che specifica dove sarà posizionato il pallino all'interno dell'ingombro impostato. I valori sono quelli mostrati sull'immagine posmarker in fondo;
- stringaCodice: la stringa contenente il codice di descrizione del marker;
- layer: su che layer è attivo il marker. Impostando il valore a -1 il marker sarà visibile su tutti i layer.

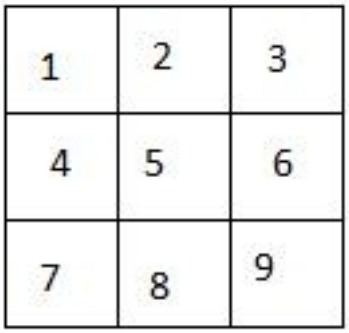


Figura 5.7: Posizione pallino marker.

All'interno della stampa:

All'interno del comando *ARG* aggiungete il parametro /MARKER come da esempio:

Esempio 24. Marker nella stampa.
tipo3D: è un parametro che permette di specificare se i pallini in stampa devono essere 3D, ovvero esso può assumere i seguenti valori:

0 : il pallino non dipende dallo zoom della stampa.

1 : il pallino viene ridimensionato in base allo zoom della camera; non è un pallino 3D, nel senso che esso non dipende dalla distanza del box dall'osservatore e quindi tutti i pallini a video hanno le medesime dimensioni.

2 : il pallino è a tutti gli effetti 3D, ovvero dipende dalla distanza del marker dall'osservatore.

Codice 204. Aggiungere marker.

```
1 ARG (0,0) (210,175) V0 /BI /MARKER
```

Se non specificate nulla all'interno del comando MARKER tutti i layer saranno stampati. Altrimenti potete specificare l'indice dei layer che desiderate.

Aggiungete la variante CODICE contenente tre righe

1 ROSSO 2 VERDE 3 BLU

Il codice della regola è in e codice 205. La tipologia 050 è una tipologia che si affianca alla finitura 21; all'interno della tipologia non vi sono parametri particolari (potete lasciare lato invisibile per questo esempio) La finitura da usare è in codice 206. La stampa PROVA_MARKER.stn porta il seguente codice 207.

Codice 205. Stampa PROVA_MARKER.stn.

```
#LETS colore = 0||255;0;0
#IF $(Codice:1) = 1
  #LETS colore = 0||255;0;0
#ENDIF
#IF $(Codice:1) = 2
  #LETS colore = 0||0;255;0
#ENDIF
#IF $(Codice:1) = 3
  #LETS colore = 0||0;0;255
#ENDIF
050,C1=$(colore),FACC=21,X1=$(Codice:2)
```

Codice 206. Stampa PROVA_MARKER.stn.

```
Sub Anta21(dl,da,dp,c1,c2,pTerra,lato,px,py,pa,pAltezza,
    pLarghezza)
  wrPannello "APL",c1,dl,da,dp
  wrMarker dl,da,dp,5,zparametro(1),-1
end sub
```

Codice 207. Stampa PROVA_MARKER.stn.

```
ORIZZONTALE
DIM 280 200
FONT TAHOMA
METRIC 1
ELAGRUPPO INIT //Contiene eventuali intestazioni di pagina-
    formato ecc
#LETV markers = $(.idmarkers)
ARG (0,0) (210,175) V0 /BI /MARKER //Questo abilita I MARKER
RESETTESTI 1
#LET nx = 5.5
#LET contegg = 1
#FOR $(markers),,codatt //Scorriamo tutti i codici presenti
    in grafica
  TX TL 12 (217,$(nx)) $(contegg)-$(codatt) //stampo il
      codice
  #LET nx=$(nx)+5.5
  #LET fl=1
  #LET contegg =$(contegg)+1
#ENDFOR
```

```
17 FONT TAHOMA
18 TX BR 25B (185,10)
```

5.11 Stampa dei listini

Per la gestione e stampa dei prezzi di listino di particolari configurazioni di prodotto[7] occorre disporre del modulo M06(Listini).

Il modulo listini abilita 3 nuove funzioni nel menu delle associazioni. Tali funzioni permettono di inserire delle combinazioni di varianti sulle quali si vuole sia fatto il calcolo del prezzo e successivamente, attivare per ogni articolo l'elenco delle combinazioni valide (nella stampa di esempio ne sono state fatte quattro).

E' possibile anche l'impostazione della suddivisione AKA, degli articoli a fronte della quale per una certa tipologia gli articoli sono raggruppabili con un criterio AKA. Viene in questo caso creato un solo codice per la gestione delle immagini e viene associata una descrizione comune a più articoli (nell'esempio allegato il codice AKA è stato fatto con il criterio che elementi di larghezza diversa dovessero essere accorpati).

Dopo queste impostazioni si passa alla creazione delle immagini ad alta qualità nella schermata della creazione delle miniature, qui possono essere elaborati sia i prezzi generati dalle varie configurazioni (memorizzati nella tabella ARTCOMB), sia le immagini di alta qualità per il listino, che seguono il criterio dei codici AKA.

[7] Disponibile dalla versione 569 di ***3CadEvolution*** .

6 Lavorazioni

La procedura ***3CadEvolution*** , oltre a fornire la grafica e le stampe connesse all'ordine, consente il passaggio dei dati alla produzione attraverso dei file per le macchine a controllo numerico. Le macchine a cui si fa riferimento in questa sede sono quelle di tipo foro inseritrice.

Anche in riferimento a questo argomento, dobbiamo operare una distinzione tra la gestione delle cucine, a cui si possono tranquillamente associare i bagni, e l'interparete. Nel primo gruppo gli elementi che necessitano di lavorazioni sono:

- i top e le mensole, per i quali è necessario conoscere i tagli iniziali, finali ed i fori per eventuali lavelli, piani cottura, cappe o faretti;
- le alzatine, conrici e zoccoli, per i quali servono i tagli;
- le ante che devono essere forate per il posizionamento delle maniglie;
- i fianchi delle colonne che hanno attrezzatura interna.

Nell'interparete le informazioni da passare sono più numerose e riguardano quasi tutti i componenti: fianchi, top, schiene, ripiani ecc.

Per operare, la macchina a controllo numerico necessita dei programmi. Questi, nella maggior parte dei casi, sono delle liste di lavoro la cui sequenza è diversa a seconda delle esigenze (per colore, per collo, per ordine, ecc.).

Meno frequente è la gestione del codice a barre, che prevede la generazione di una etichetta per ogni pezzo. In tal caso la macchina esegue le lavorazioni in base alle informazioni contenute nel codice.

Le regole di foratura consentono la produzione del codice di controllo per tutte le macchine collegate alla lavorazione dei pezzi di un mobile.

Lo schema generale di funzionamento del processo di generazione dei programmi di foratura può essere sintetizzato dal seguente diagramma, Figura 6.1 nella pagina 236, che evidenzia inoltre il ruolo giocato dai vari tipi di file.

Il diagramma può essere così commentato: dato un insieme di regole che si occupano di definire i metodi di lavorazione per una certa tipologia di componente e dato un numero indefinito di tali componenti, inseriti nell'ambito di una composizione, andiamo a verificare quali elementi provocano lavorazione su di essi. Una volta riscontrata la necessità di lavorazione tra due elementi, le regole di foratura producono codice (a livello intermedio) che tiene conto delle dimensioni dei due pezzi, del punto (o dell'area) su cui intervenire, del lato di lavorazione, del tipo di utensile da usare e così di seguito.

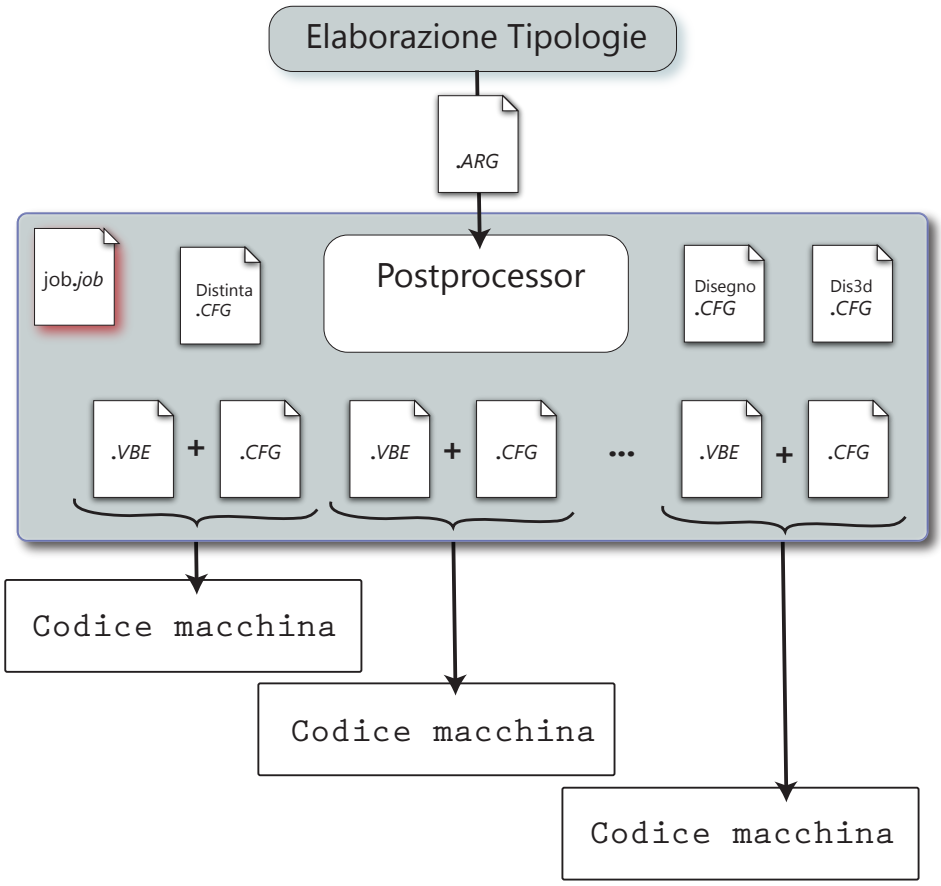


Figura 6.1: Schema funzionamento job.

Il codice prodotto da questa prima fase di elaborazione è in grado di governare qualsiasi macchina utensile se esiste il traduttore intermedio appropriato. Il traduttore riceve in ingresso il file *.ARG*, prodotto dalla prima fase, e il file di configurazione della macchina, il file *.CFG*. Dopo questa seconda fase è pronto in uscita il codice finale comprensibile dalla macchina utensile.

Il file *job.job*: si trova nella directory contenente i file *.VBS* e contiene l'elenco degli script definendo la sequenza d'esecuzione di essi.

Il file *Disegno.cfg*: definisce il disegno della lavorazione che viene visualizzato nella scheda Disegno.

6.1 Premesse

Per essere abilitato alle lavorazioni la chiave hardware necessita del modulo M12. Prima di vedere nel dettaglio le voci specifiche della finestra lavorazioni è bene fare una premessa:

Se si sviluppa un catalogo con l'intenzione di svilupparne anche i job, è doveroso mantenere un criterio tra dimensioni anagrafiche dei componenti, orientamento della tipologia, e job, spieghiamo meglio questo concetto:

supponiamo di trattare un fianco un fondo e un'anta; graficamente questi oggetti sono messi rispettivamente sui piani YZ, XZ e YX. Dimensionalmente parlando il fianco avrà spessore nel senso della larghezza, il fondo avrà lo spessore nel senso dell'altezza e l'anta avrà lo spessore nel senso della profondità.

Se avessimo una distinta avremmo una tabella come in Figura 6.2.

Codice	Descrizione	UM	C/F	PZ	Qtà	Dimensioni	Varianti	Nota1	Nota2	Nota3	Nota4	Nota5	Nota6	FM	Matr
ANTA	Anta				1	600x700x20									#M2/4
COPERCHIO	coperchio				1	540x30x500									#M2/2
FIANCODX	fiancodx				1	30x700x500									#M2/1
FIANCOSX	fiancosx				1	30x700x500									#M2/0
FONDO	fondo				1	540x30x500									#M2/3

Figura 6.2: Orientamento delle tipologie per il job.

Come notiamo le dimensioni non sono ordinate tra di loro ma, altezza, larghezza e spessore, sono alternativamente nella prima, seconda o terza posizione.

È quindi utile avere una anagrafica dei componenti ordinata, avendo cura di tenere alla prima misura, l'altezza del componente, alla seconda misura la larghezza, e per ultima lo spessore.

In modo da avere una lista come in Figura 6.3 nella pagina 238.

Per quanto riguarda l'orientamento della tipologia facciamo affidamento a l'interfaccia visibile in Figura 6.4 nella pagina 238.

Codice	Descrizione	UM	C/F	PZ	Qtà	Dimensioni	Varianti	Nota1	Nota2	Nota3	Nota4	Nota5	Nota6	FM	Matr
ANTA	Anta				1	700x600x20									#M2/4
COPERCHIO	coperchio				1	540x500x30									#M2/2
FIANCODX	fiancodx				1	700x500x30									#M2/1
FIANCOSX	fiancosx				1	700x500x30									#M2/0
FONDO	fondo				1	540x500x30									#M2/3

Figura 6.3: Orientamento delle tipologie per il job.

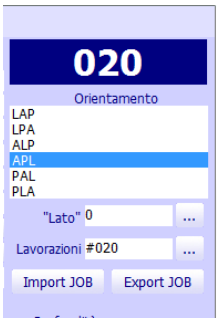


Figura 6.4: Orientamento delle tipologie.

Di fatto dettiamo un ordine di importanza per le tre dimensioni, con un esempio possiamo dire che:

Il fianco ha orientamento *APL*.

Il fondo ha orientamento *LPA*.

L'anta ha orientamento *ALP*.

Ne deduciamo che questo tipo di impostazioni ne risulta coerente con l'orientamento anagrafico: ovvero la prima dimensione ne è l'altezza la seconda la larghezza e la terza lo spessore.

6.2 Interfaccia

Le lavorazioni vengono create dalle tipologie, accediamo quindi dapprima all'interfaccia della tipologia, e facciamo click sul tasto ..., in Figura 6.5 nella pagina 239.

Si apre la finestra in Figura 6.6 nella pagina 239.

Vediamo nello specifico i vari ambienti.

Self-ini

In questo ambiente vengono dichiarate tutte le lavorazioni che necessitano di essere eseguite per prime, solitamente si tiene questo blocco come luogo di dichiarazioni di variabili o estrapolazione dei valori dai parametri.

Lavorazioni per contatto

In questa schermata, Figura 6.7 nella pagina 240, nella colonna tipologia possiamo dichiarare tutte le tipologie che, una volta a contatto con la tipologia corrente creeranno una lavorazione, nella colonna lato invece, dichiariamo in quale lato la tipologia corrente dovrà essere sensibile alla tipologia che porta la lavorazione.

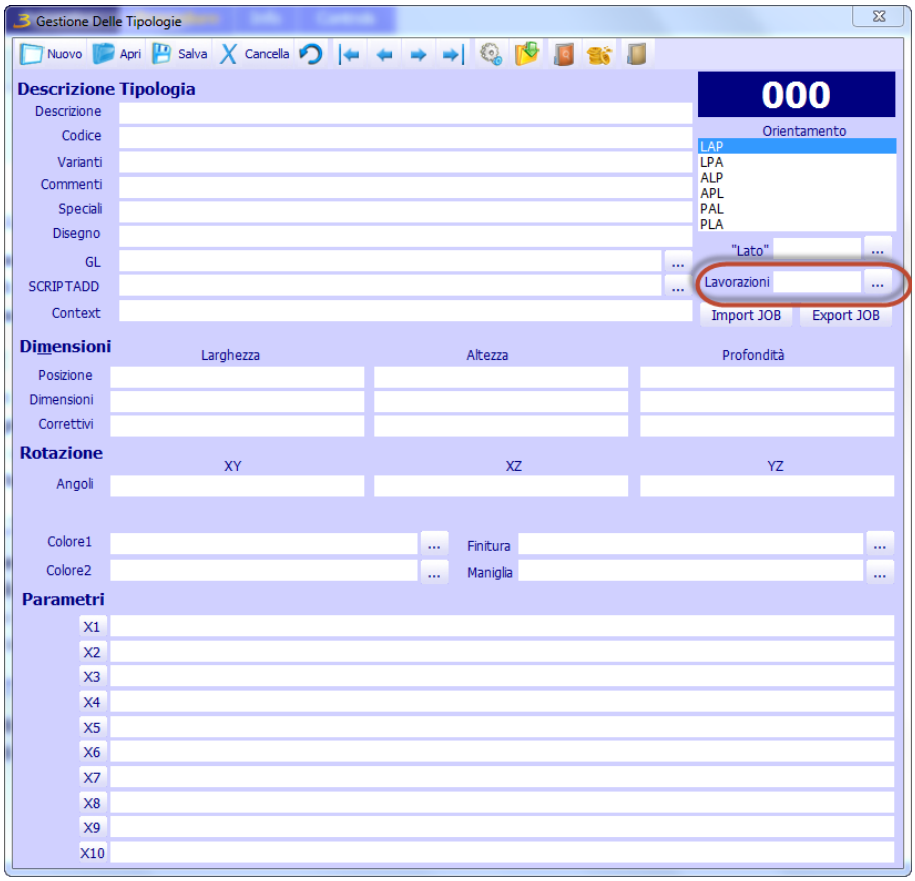


Figura 6.5: Interfaccia della tipologia.

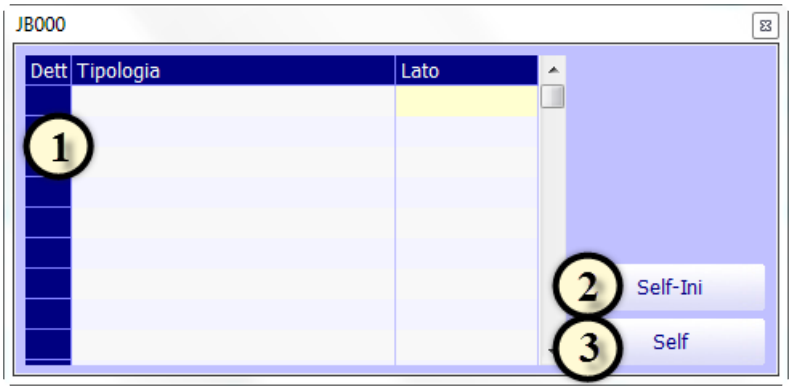


Figura 6.6: Ambienti per le tipologie di lavorazione.

(1) *Lavorazioni per contatto.*
(2) *Lavorazioni iniziali.*
(3) *Lavorazioni finali.*

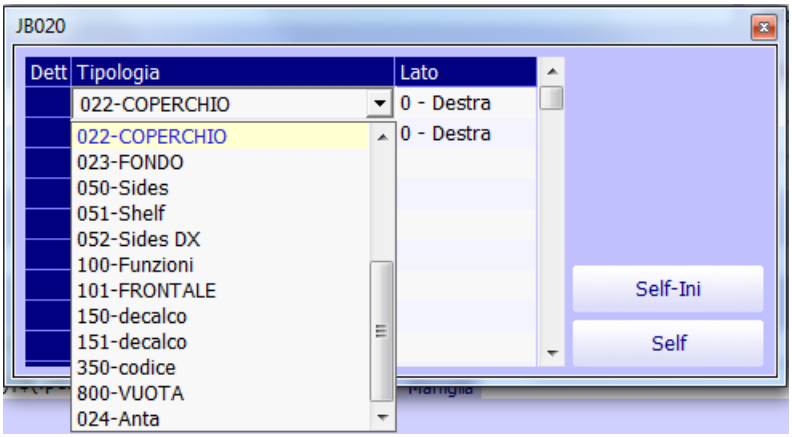


Figura 6.7: Dichiarazione delle tipologie a contatto.

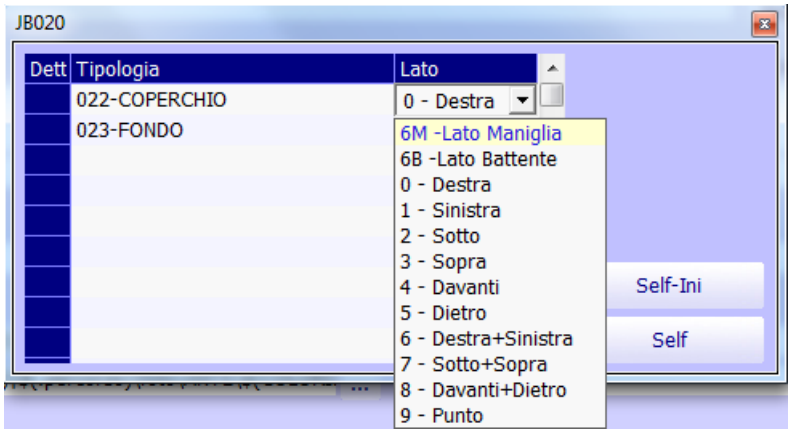


Figura 6.8: Dichiarazione del lato sensibile per la lavorazione.

In questo esempio in Figura 6.8 nella pagina 240 stiamo trattando un fianco sinistro, quindi coperchio e fondo porteranno le lavorazioni per contatto sul lato destro del fianco.

Self

In questo ambiente dichiariamo tutte le lavorazioni che dovranno essere eseguite per ultime.

6.3 Variabili

Nei vari ambienti, self-ini, self, e lavorazioni per contatto, abbiamo a disposizione una serie di variabili di sistema.

6.3.1 Variabili della tipologia corrente

Disponibili in tutti e tre gli ambienti:

- $(XL), $(XA), $(XP): sono le dimensioni della tipologia (componente), e sono lette direttamente dai campi dimensionali della tipologia;
- $(XX1), $(XX2), $(XX...), $(XX10): sono i parametri della tipologia. Di fatto nell'ambiente job, come nell'ambiente distinta/listino a formule, e script finiture, non sono presenti le variabili e le varianti della grafica, quindi ogni qualvolta necessitiamo di un valore dalla grafica, lo dobbiamo dichiarare in un parametro, richiamando la variabile $(XX...), di fatto abbiamo a disposizione tutto il parametro, ne consegue che lo dobbiamo estrapolare con la seguente funzione:

  Esempio 1. Estrapolazione del parametro della tipologia dalla grafica. La variabile $(par1val2) assumerà il valore pluto.

  X1 pippo/pluto/minni

  Codice 208. .]Utilizzo della funzione $[ex].

```
1   #LETS par1val2=$[ex;$(XX1);,;2]
```

Queste dichiarazioni è bene scriverle nel self-ini.

6.3.2 Variabili della tipologia appoggiata

Disponibili solo nell'ambiente lavorazioni per contatto.

- $(DL), $(DA), $(DP): sono le dimensioni della tipologia (componente), e sono lette direttamente dai campi dimensionali della tipologia;
- $(PL), $(PA), $(PP): sono i posizionamenti della tipologia relativi alla tipologia corrente, vedere il disegno di seguito come esempio;
- $(XP1), $(XP2), $(XP...), $(XP10): sono i parametri della tipologia, anche in questo caso l'estrapolazione è affidata alla funzione $[ex;].

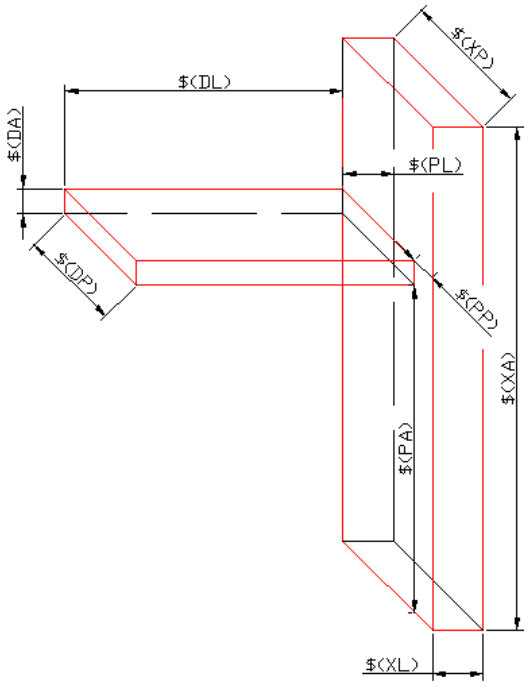
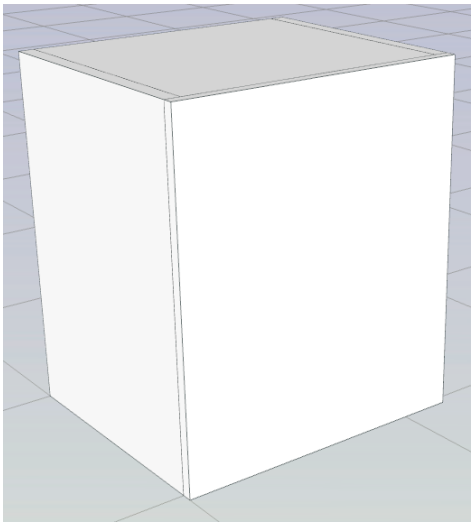


Figura 6.9: Schema delle variabili della tipologia appogiata.

6.4 Creazione della lavorazione

Creiamo un contenitore semplice composto da 2 fianchi un coperchio un fondo e un'anta, come in Figura 6.10 nella pagina 242.



Utiliziamo le tipologie:

20 fianco sinistro

21 fianco destro

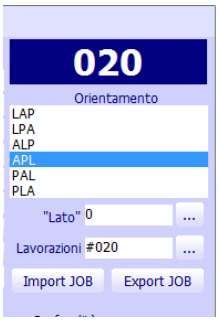
22 coperchio

23 fondo

24 anta

Figura 6.10: Box elementare.

Da qui creiamo subito le lavorazioni per contatto con coperchio e fondo, Figura 6.11 nella pagina 243.



Apriamo la finestra della tipologia ed entriamo nell'ambiente job, notiamo che il programma crea il nome della lavorazione con la seguente sintassi #020, dove 020 è il nome della tipologia.



Figura 6.11: Lavorazioni per contatto, coperchio e fondo.

Cliccando sul rettangolo blu (evidenziato), apriamo l'ambiente come in Figura 6.12 per la scrittura delle regole di foratura, tra la tipologia corrente e quella per contatto.

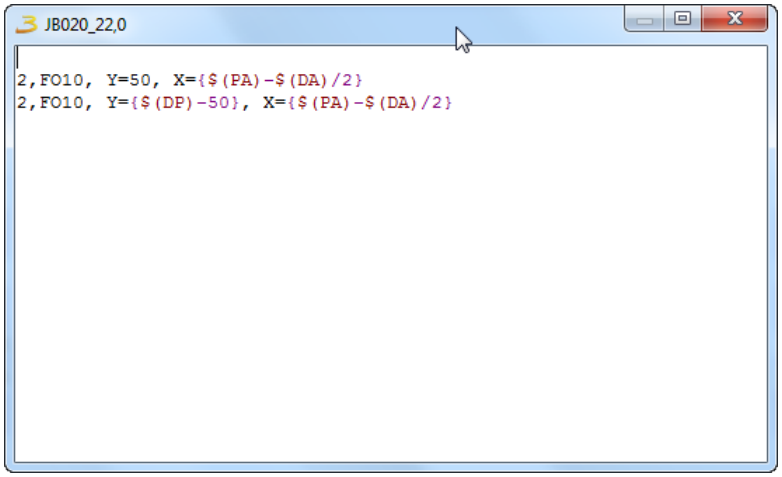


Figura 6.12: Editor per la regola di foratura.

Come già notiamo nella barra del titolo, stiamo eseguendo le lavorazioni nella tipologia 20, causate dal contatto con la tipologia 22, nel lato 0, ovvero a destra.

Analizziamo la riga di dichiarazione:

2 : è la faccia sulla quale si esegue la lavorazione, nel caso in esempio è opzionale in quanto il programma riconosce automaticamente la faccia, tuttavia, in alcuni casi è necessario dichiararla, in quanto la stessa tipologia a contatto porta lavorazioni non necessariamente sulla faccia in questione, basti pensare le lavorazioni di un anta su un fianco, l'anta è appoggiata sul bordo del fianco ma le lavorazioni sono necessarie sul lato interno dello stesso.

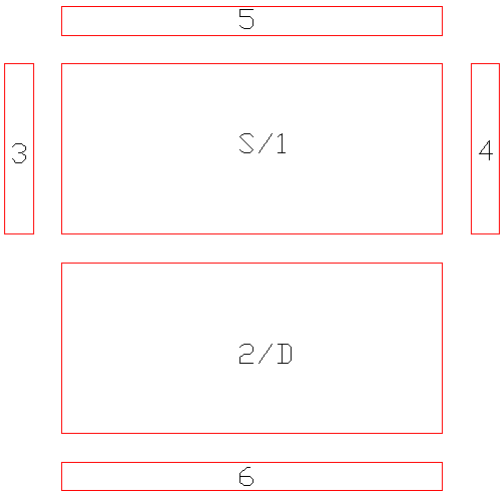


Figura 6.13: Schema di spiegazione di disposizione delle facce per le lavorazioni.

FO10 : è il nome della lavorazione, solitamente è consigliabile utilizzare dei nomi mnemonici, anche se di fantasia, in questo caso stiamo eseguendo un foro diametro 10mm.

X, Y : sono le coordinate delle lavorazioni.

Per visualizzare quanto scritto creiamo un ordine con il contenitore in questione e lo salviamo, poi ci spostiamo in Procedure->Elaborazioni->Job.

Fino ad arrivare alla finestra in Figura 6.14 nella pagina 245.

Tralasciando momentaneamente il significato di tutti i pulsanti presenti, selezioniamo il nostro ordine dalla lista di sinistra con un click, facciamo attenzione che siano spuntate le opzioni cancella elaborazioni precedenti e genera programmi, a questo punto facciamo click su Elabora.

A questo punto ci spostiamo sulla seconda tab in alto Disegno. Selezioniamo con il puntatore, la cartella temporanea interessata, in alto a sinistra, poi selezioniamo la lavorazione, a sinistra in centro, quello che ci appare, è il disegno con le lavorazioni create nella tipologia come in Figura 6.15 nella pagina 245.

6.5 Elaborazioni

Vista nel dettaglio l'interfaccia delle elaborazioni presenta sulla sinistra tutta la parte di selezione ordini/carichi da elaborare, nella parte centrale sono visibili gli ordini in elaborazione, e nella parte sotto l'avanzamento delle elaborazioni.

Nella parte di destra notiamo invece le opzioni attivabili per l'elaborazione e la dichiarazione della cartella nella quale verranno salvati i file elaborati (.ARG) .

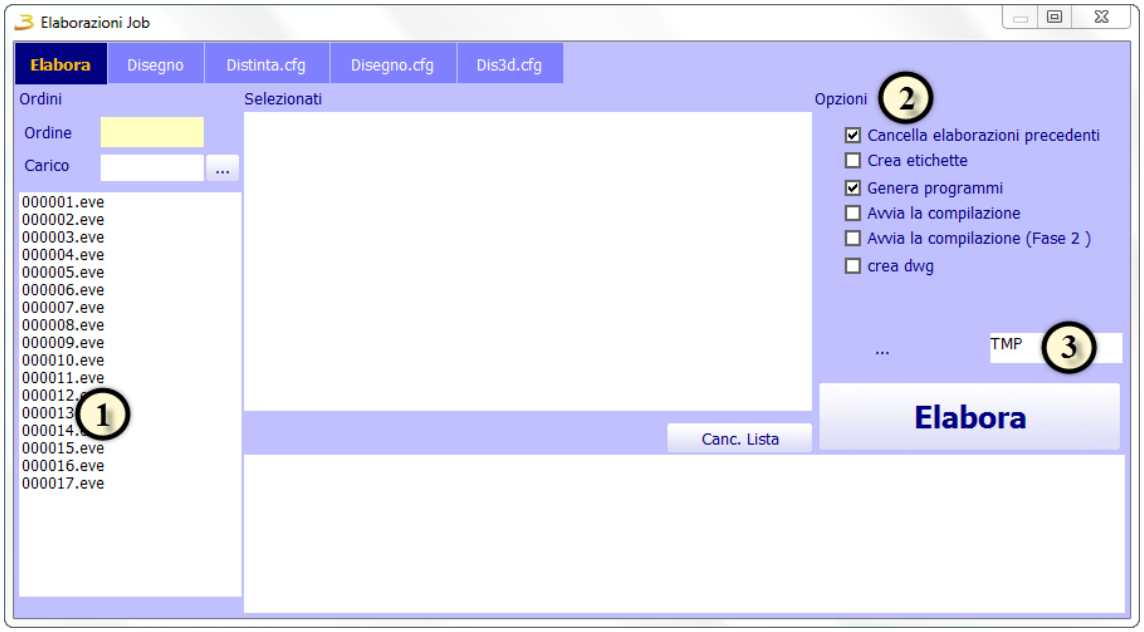


Figura 6.14: Elaborazioni.

(1) *Lista ordini.*
(2) *Opzioni.*
(3) *Cartella temporanea.*

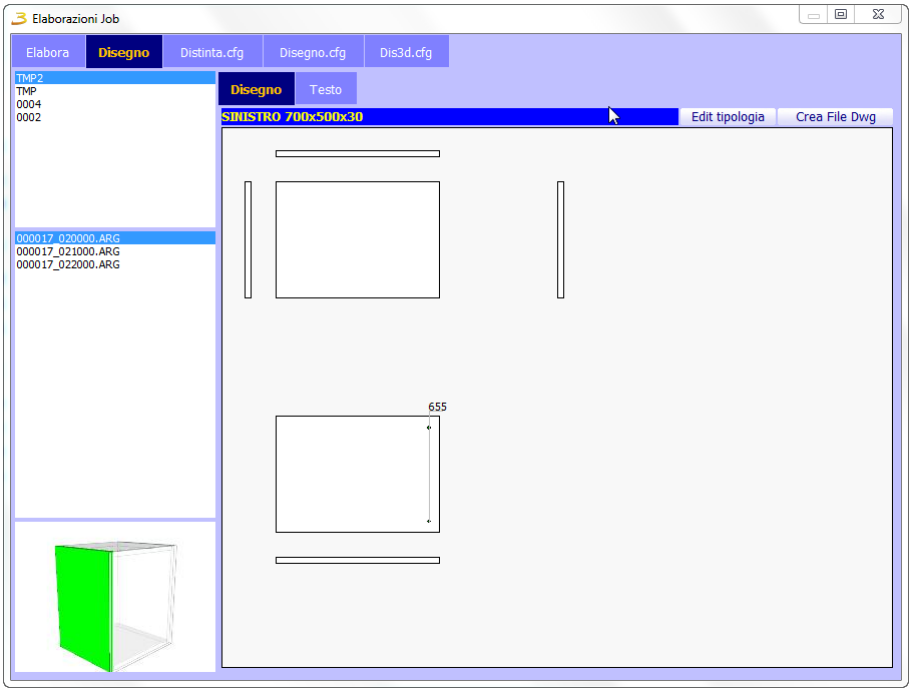


Figura 6.15: Disegno della lavorazione.

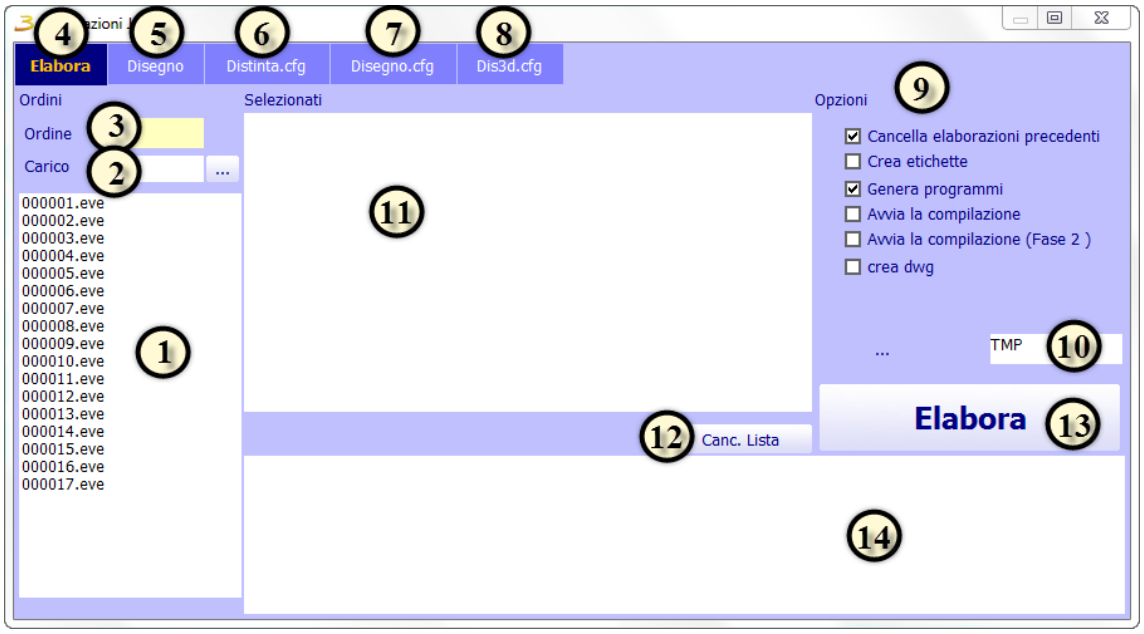


Figura 6.16: Elaborazioni schermata generale.

(1) *Lista ordini.*
(2) *Inserimento carico.*
(3) *Inserimento ordine.*
(4) *Tab elaborazioni.*
(5) *Tab disegno.*
(6) *Tab distinta.cfg .*
(7) *Tab disegno.cfg .*
(8) *Tab dis3d.cfg .*
(9) *Opzioni.*
(10) *Cartella temporanea Nella quale vengono salvati gli ARG elaborati .*
(11) *Ordini in elaborazione.*
(12) *Cancella la lista degli ordini in elaborazione.*
(13) *Avvia l'elaborazione.*
(14) *Avanzamento elaborazioni.*

Nel dettaglio vediamo le voci:

- cancella elaborazioni precedenti: svuota la cartella (in questo caso `TMP`) di tutti i file delle elaborazioni precedenti;
- crea etichette: genera le immagini dei colli nella cartella `ETI` in _ECAD-PRO.
- genera programmi: crea i file .ARG con le informazioni delle lavorazioni
- avvia la compilazione: elabora i file .ARG secondo i post processor dichiarati nel file *job.job*;
- avvia la compilazione (fase 2): come sopra ma secondo i post processor dichiarati nel file *job1.job*, serve per alcune macchine che richiedono la doppia compilazione;
- crea dwg: crea i disegni delle lavorazioni in formato *DWG*.

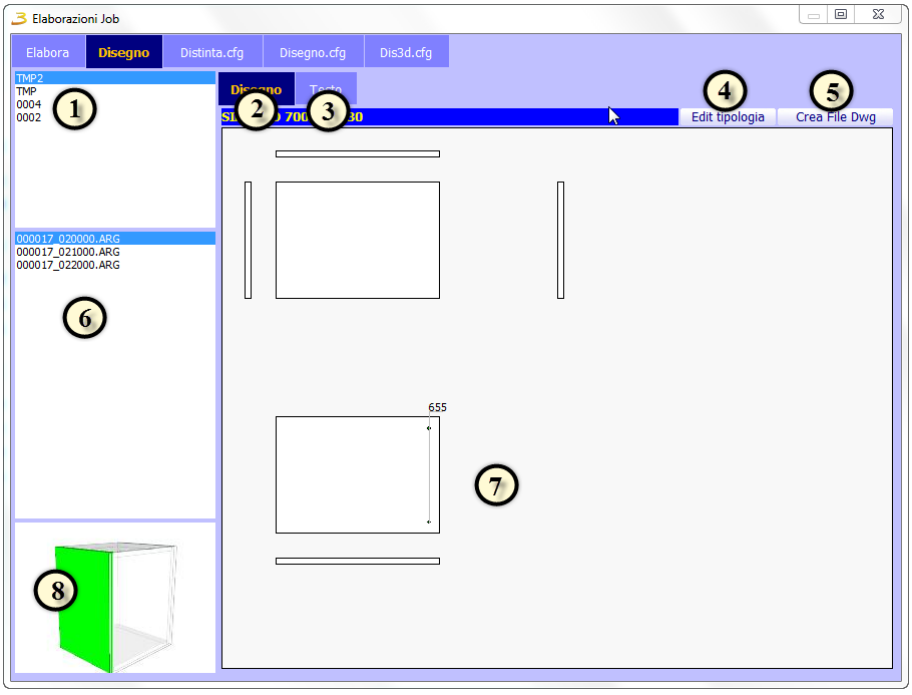


Figura 6.17: Schermata del disegno della lavorazione.

(1) *Lista cartelle contenenti le lavorazioni.*
(2) *Visualizzazione disegno.*
(3) *Visualizzazione testo (lavorazioni e coordinate).*
(4) *Apre la finestra di selezione delle tipologie.*
(5) *Crea il disegno in formato . DWG del componente corrente.*
(6) *Lista delle lavorazioni (.ARG).*
(7) *Anteprima o in formato disegno o in formato testo.*
(8) *Anteprima del componente elaborato (.arg).*

La tab disegno ci si presenta con a sinistra la lista di cartelle nelle quali ci sono i file elaborati, il contenuto di tali cartelle e l'anteprima del componente elaborato.

La parte centrale e adibita alla visualizzazione del disegno/testo del componente elaborato, in più nella parte in alto a destra ci sono i pulsanti per richiamare ed editare direttamente la tipologia, e per creare il disegno in formato . *DWG*.

6.6 File

6.6.1 File *Distinta.CFG*

In questo script è possibile creare la lista dei componenti-ferramenta, generati dalle lavorazioni, intendiamo se sta eseguendo un foro per spina, è possibile decidere da qui, che una volta eseguito il foro, venga salvato in distinta anche il codice della spina.

Vediamo in che modo:

Creare il gruppo con lo stesso nome della lavorazione.

Codice 209. Lista di elaborazioni.

```
[FO10]
Codicespina,1
```

Il codice *codicespina* è salvato direttamente nella tabella JOBDISTINTA di ECADMASTER.

6.6.2 File *Disegno.CFG*

Altro non serve che per ridisegnare quello che il programma visualizza in automatico, reinterpretando le varie lavorazioni con dei disegni differenti.

La struttura del file è così composta:

Codice 210. Struttura del file *Disegno.CFG* .

```
[MACRO]
[LAVORAZIONE1]
[LAVORAZIONE2]
[LAVORAZIONE...]
```

Le lavorazioni sono i nomi delle lavorazioni che abbiamo usato nell'ambiente job nelle tipologie, utilizzando l'esempio della sottosezione Creazione della lavorazione, nel codice 211 è possibile leggere un file disegno valido.

Codice 211. File *Disegno.CFG* .

```
[MACRO]
[FO10] // nome della lavorazione dichiarata nella tipologia
// FORO TEMPORANEO SOLO GRAFICA
FO,2;1,$(x)-5,$(y)-5,$(x)+5,$(y)+5 // disegno un cerchio di
    diametro 10mm
QX,$(x),$(y)
Qy,$(x),$(y)
```

Notiamo subito che è possibile utilizzare il commento // come nell'ambiente regole.

Vediamo nel dettaglio la sintassi:

FO : disegna un cerchio pieno;

2 : uso il colore verde;

1 : uso lo stile di linea tratteggi larghi;

X, Y : sono le coordinate del centro dalla lavorazione derivata dalle tipologie, si noti che per avere un cerchio è necessario discostarsi dalla coordina, del centro verso i 4 quadranti.

La sintassi da usare è la seguente:

<nome>, <Colore>; <pTipo>; <pColore>, <p1x>, <p1y>, <p2x>, <p2y>
...

Colore	
0	Nero
1	Blue
2	Verde
3	Verdemare
4	Rosso
5	Viola
6	Fucsia
9	ciano

Tabella 6.1: Colori per il file *Disegno.CFG* .

pTipo	
0	Linea
1	Tratteggi larghi
2	Tratteggi stretti

Tabella 6.2 : continua nella prossima pagina

Tabella 6.2 : continua dalla pagina precedente

pTipo	
3	Asse linea lineetta
4	Linea 2 lineette

Tabella 6.2: Stile di linea per il file *Disegno.CFG* .

Oggetto	
BX	Rettangolo
FO	Foro
B2	Rettangolo Vuoto
BV	Rettangolo
CE	Cerchio
AR	Arco Cerchio (4 punti)
LI	Linea
PO	Poligono
PL	PolyLinea
QX	Quota X
QY	Quota Y
TX	Testo

Tabella 6.3: Oggetti da disegnare per il file *Disegno.CFG* .

6.6.3 File *Dis3d.CFG.*

Consente di creare degli oggetti per simulare la ferramenta durante l'esplosione dell'articolo.

Lo script consiste nel richiamare a sua volta un file di tipo .xp (file script).

La sintassi per visualizzare una grafica come in Figura 6.18 nella pagina 251, è visibile in 212.

Codice 212. Inserimento ferramenta durante l'esplosione.

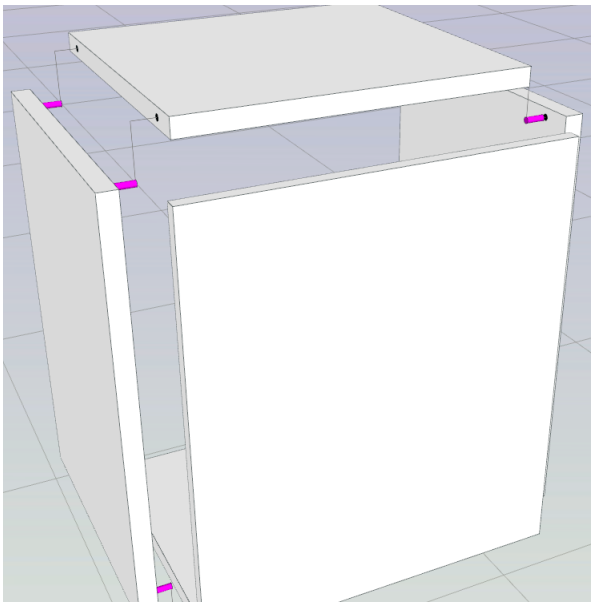


Figura 6.18: Articolo con ferramenta durante l'esplosione.

```
[FO10] \\richiamo la lavorazione come dichiarata nella
    tipologia
0,$(X),$(Y),0 ,_ecadpro\matbase\foro.xp;5;1;1
0,$(X),$(Y),1 ,_ecadpro\matbase\foro.xp;5;1;1
0,$(X),$(Y),0.3,_ecadpro\matbase\foro.xp;5;30;6
```

Vediamo nel dettaglio il significato dei vari campi:

- 0 flag di indicazione: o numero di faccia se diversa da default o simbolo + per indicare che la lavorazione non va raccordata con pannello incidente;
- 1, 2: posizioni X e Y;
- 0.3 coefficiente da 0 a 1: 0 la lavorazione è appoggiata al pannello incidente e 1 a quello inciso, valgono tutti i valori intermedi;
- 4 fileXP: contenente la lavorazione, diviso in campi separati da `;` i campi sopra individuano i valori da inserire nei parametri dello script.

6.7 Postprocessor

Il post processor non è altro che un programma (script in VBS), che permette di decodificare.

Il programma scritto nel file .ARG, nel linguaggio comprensibile al cnc.

Il file che solitamente ha estensione VBS, va dichiarato nel file *job.job*, unitamente al file .VBS, solitamente è anche un file .BAT, che si occupa di spostare/copiare, i file tradotti dall'ARG nella cartella di lettura del cnc.

Nel file *job.job*, ci saranno VBS e BAT tanti quanti sono le macchine presenti nell'azienda.

Unitamente al post processor esiste anche un corrispondente file di configurazione con estensione .CFG, in tale file verranno dichiarate tutte le lavorazioni presenti nei job delle tipologie e per ogni lavorazione verrà dichiarata la rispettiva sintassi della macchina:

Tornando all'esempio di foratura nella sotto sezione Creazione della lavorazione e supponendo di avere un cnc Albatros.

Avremmo:

```
Albatros.vbs
Albatros.bat
Albatros.cfg
```

I primi due dichiarati nel *job.job*, l'ultimo (tra l'altro visibile nella finestra delle elaborazioni accanto alle tab disegno, distinta e dis3d), avrà un aspetto come segue in codice 213.

Codice 213. File di configurazione .CFG.

```
[MACCHINE]
2
JOBAALB1
JOBAALB2


[MACRO]
FC002




[FC002]

  1:1:W#81:WTP WS=1,#1=$(X) #2=$(Y) #3=-15 #1002=8 #8015=0
     #2005=3 #2002=6000 #1001=1 #90=MEZZERIA
  2:1:W#81:WTP WS=1,#1=$(X) #2=$(Y) #3=-15 #1002=8 #8015=0
     #2005=3 #2002=6000 #1001=1 #90=MEZZERIA
```

[MACCHINE]: Numero di macchine e configurazione della cartella di destinazione;

[MACRO]: lavorazione come dichiarata nelle tipologie;

[FC002]: traduzione per il cnc [8].

6.8 Lavorazioni tra componenti a riconduzione

Un caso particolare di lavorazione si può avere tra componenti nei box e componenti a riconduzione, come ad esempio delle boiserie accostate e una pedana al metro lineare come nel disegno di esempio in Figura 6.19 nella pagina 253.

[8] Si noti 2 righe una per ogni macchina.

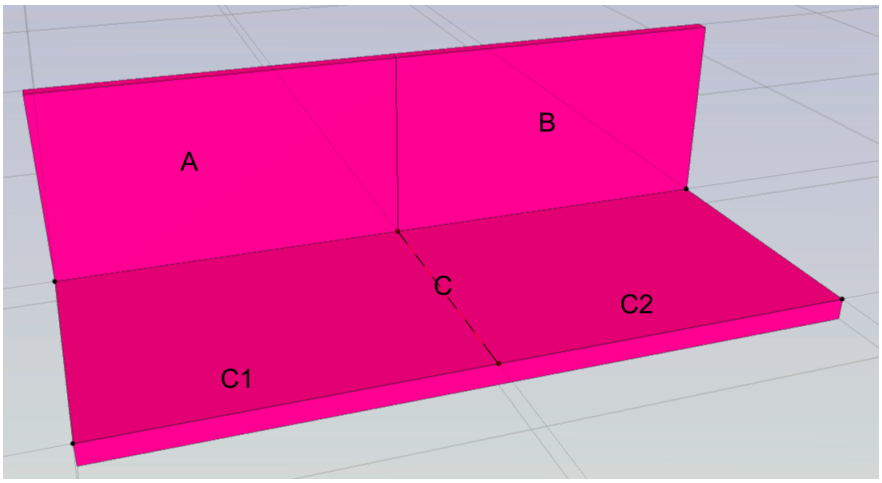


Figura 6.19: Boiserie accostate e una pedana al metro lineare, mobile che necessita la lavorazione a riconduzione.

Supponiamo che i pannelli A e B ricevano delle lavorazioni dall'elemento a riconduzione C,formato dalla giunzione delle 2 tipologie C1 e C2.

Quindi nella dichiarazione delle lavorazioni per contatto della tipologia A richiameremo la tipologia Cn, ma come facciamo a sapere se stiamo lavorando con la tipologia singola o quella a riconduzione?

Il sistema mette a disposizione una variabile *$(_isTop)* che vale:

0 : nel caso della tipologia singola;

1 : nel caso della tipologia ricondotta.

Supponiamo che i due articoli inseriti abbiano larghezza 500 mm, se rilasciamo come esempio codice 214.

Codice 214. Il contenuto della variabile $(_isTop).

```
#MSG $(_isTop) ** $(DL)
```

Riceveremo 2 messaggi consecutivi con i seguenti valori:

0 ** 500 e successivamente 1 ** 1000

A questo punto possiamo utilizzare la variabile e i valori come meglio crediamo ad esempio in un costrutto *#IF* come in codice 215.

Codice 215. Utilizzo di &(_isTop) nel *#IF*.

```
\#IF $(_isTop)=1 , ecc, ecc...
```

7 Gestione degli imballi

7.1 Interfaccia

Spostandoci in Procedure->Anagrafica->Imballi, ci si presenta la schermata come in Figura 7.1.

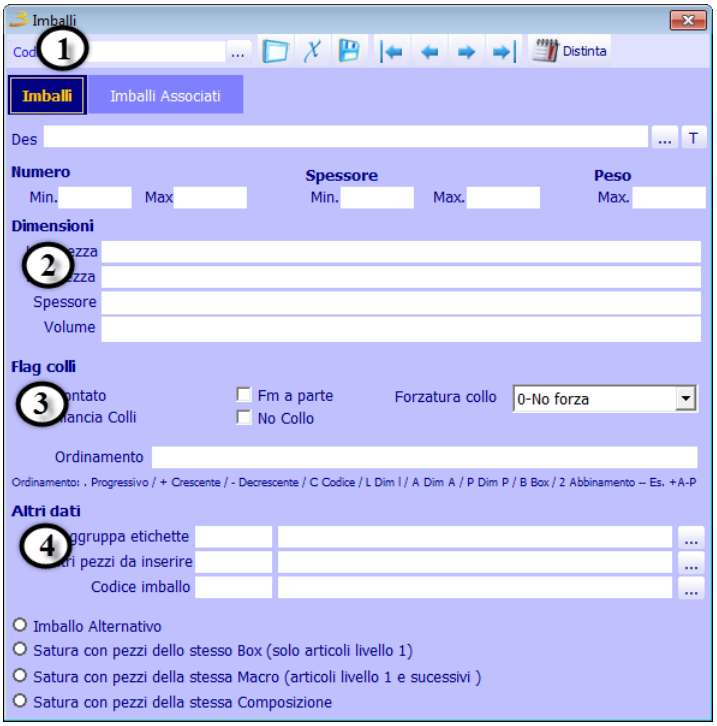


Figura 7.1: Interfaccia imballi.

(1) *Intestazione.*
(2) *Caratteristiche.*
(3) *Flag colli.*
(3) *Altri dati.*

Andiamo ora a vedere nel dettaglio il significato di tutte le funzionalità che ci sono messe a disposizione.

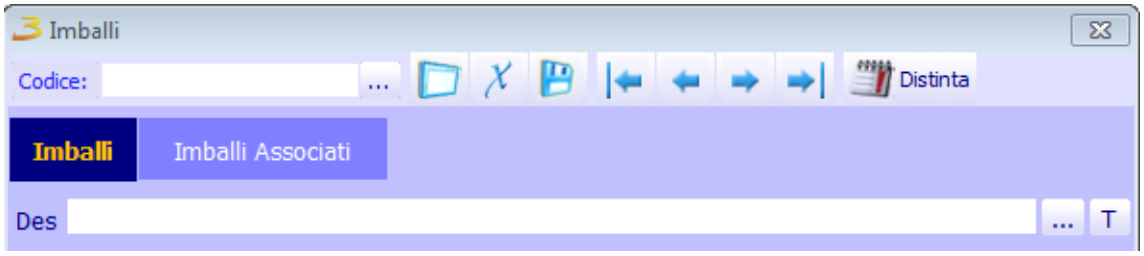


Figura 7.2: Interfaccia imballi, intestazione.

Intestazione:

- *Codice*: inserisce il codice imballo, coincide, con la creazione, o il caricamento dell'imballo interessato. Il codice imballo è un identificativo e può avere caratteri alfanumerici, deve essere unico e preferibilmente con un nome corto.
- *Des*: è una breve descrizione per identificare l'imballo.
- *Imballi*: è la tabella (tab) per descrivere e caratterizzare l'imballo.
- *Imballi Associati*: è la tabella (tab) per associare per accedere alle caratteristiche degli imballi associati.
- *Nuovo*: prepara l'interfaccia per la creazione di un nuovo imballo.
- *Cancella*: cancella l'imballo corrente.
- *Salva*: salva le modifiche o l'imballo appena creato.
- *Naviga*: rispettivamente le frecce eseguono i seguenti comandi

	va al primo imballo della lista
	va all'imballo precedente, secondo l'ordine alfabetico del codice
	va all'imballo successivo, secondo l'ordine alfabetico del codice
	va all'ultimo imballo della lista

- *Distinta*: accede all'interfaccia per la distinta dei componenti (cartoni, protezioni, ecc.) scaricati dall'imballo.

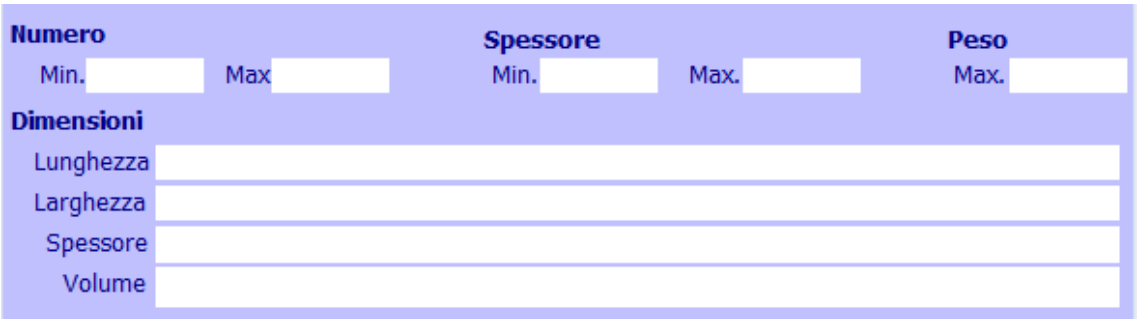


Figura 7.3: Interfaccia imballi, caratteristiche.

Caratteristiche:

In questa parte dell'interfaccia andremo a dimensionare il collo e a dichiarare il numero. Se lasciate libere l'imballo calcolerà le dimensioni e il volume in base all'articolo, tenendo conto delle dimensioni che leggiamo in preventivo, Figura 7.4 nella pagina 256.

Riga	Liv	Codice	Descrizione	Qtà	Pr.Base	Magg1	Magg2	Dimensioni	FSP	Matr.
1	1	**999_BANTA**	base con anta	1,00	0,00	0,00	0,0	600x720x560	___	990001
2	1	**TOP**	TOP	1,00	0,00	0,00	0,00	601x30x600	M___	525000
3	1	**ZOC**	ZOCCOLO Nr.1 Lung.601 (601)	1,00	0,00	0,00	0,00	601x120x10	___	532000

Figura 7.4: Calcolo delle dimensioni e del volume nel imballo.

I dati riguardanti le caratteristiche del collo vengono salvate, nel momento della conferma/salvataggio dell'ordine, nella tabella di ECADMASTER chiamata *BORDINE*, come si vede dalla Figura 7.5 nella pagina 256.

	PERCORSO	NUMERO	ETI	REPARTO	CARICO	SCARICO	CODIMB	DES	DIM1	DIM2	DIM3	VOLUME	TESTATA	VARIANTI	MATRICOLE
▶	CUCINA	257	1			0	IMB:1	IMBALLO SINGOLO	600	720	560	0,2418999969959259	False		*990001*
	CUCINA	257	2			0	IMB:1	IMBALLO SINGOLO	60	12	10	0,0006999999730740555	False		*532000*
	CUCINA	257	3	030		0	TOP:1	ELEMENTI LINEARI	601	30	600	0,010800000028014183	False		*525000*

Figura 7.5: Calcolo delle dimensioni e del volume nel imballo.

Tornando all'interfaccia per caratterizzare le dimensioni, è bene sapere come compilare i campi, e quale variabili sono messe a disposizione.

Utilizziamo la variabile:

L : per identificare la lunghezza del componente;

A : per identificare lo spessore del componente;

P : per identificare la profondità del componente;

N : per identificare il numero di componenti contenuti nell'imballo.

A tale proposito è doveroso aprire una parentesi per raccomandare all'utilizzatore di ordinare bene le dimensioni dei componenti in distinta base, nel caso di mobili smontati, con le dimensioni ordinate, in modo da trovarci lunghezza, larghezza e spessore sempre negli stessi campi di ricerca, come potete vedere in questo esempio.

Esempio 1. Ordinamento dei componenti in distinta base.

TUBO APPENDIABITI				1	448x22x15
FIANCO SOGG. SX				1	1280x560x18
FIANCO SOGG. DX				1	1280x560x18
COP_BASE JOLLY				1	463x560x18
COP_BASE JOLLY				1	463x560x18
RIP. JOLLY STRUTTURA				1	463x544x22
ANTA				1	956x497x20
SCHIENA x CONTENITORE da 500				1	1255x474x5
CASSETTO				1	497x500x156
CASSETTO				1	497x500x156

In questo caso siamo sicuri che utilizzando le variabili *LxAxP*, nel caso ad esempio del fianco avremo rispettivamente le seguenti misure *1280x560x18*.

A questo punto risulterà facile gestire, gli imballi per i componenti ad esempio accoppiati, come nel caso di 2 fianchi. Potremmo quindi compilare la tabella per il calcolo dimensionale nel seguente modo, Figura 7.6.

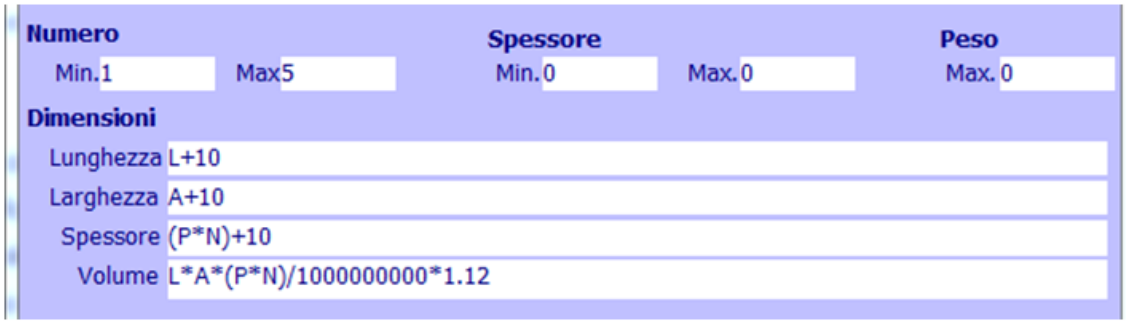


Figura 7.6: Tabella per il calcolo delle dimensioni.

Nello specifico quindi leggiamo che questo imballo: gestirà da un minimo di 1 pezzo ad un massimo di 5 pezzi, senza limite di spessore e senza limite di peso, è stato inoltre incrementato di 10mm (un ipotetico spessore per uno scatolone) l'ingombro per ogni dimensione; si noti come lo spessore tiene conto anche del numero di pezzi, incorporati nell'imballo stesso. In fine si noti come il volume può essere ricalcolato con un ulteriore aumento ad esempio del 12

7.1.1 Flag colli


Flag colli
Montato
Bilancia Colli
Fm a parte
No Collo
Forzatura collo
0-No forza
Ordinamento
Ordinamento: . Progressivo / + Crescente / - Decrescente / C Codice / L Dim l / A Dim A / P Dim P / B Box / 2 Abbinamento -- Es. +A-P

Figura 7.7: Colli.

Vediamo ora nello specifico come personalizzare, o come gestire casi speciali in imballi cosiddetti standard.

Nello specifico vediamo ogni flag a cosa corrisponde e come fa cambiare la destinazione di un componente rispetto all'imballo assegnato:

Montato

Dichiarare un componente come montato, significa che tutti i componenti di una macro (articolo composto da più componenti), faranno parte di un unico collo, quindi una unica bindella, ad esempio.

Esempio 2. Dichiarazione di un componente come montato.
Abbiamo 2 fianchi, coperchio, fondo e ripiano interno, i 2 fianchi hanno un imballo accoppiato, mentre coperchio, fondo e ripiano hanno un imballo triplo, dichiariamo inoltre un ulteriore imballo con flag montato spuntato per la schiena, questo comporterà che l'imballo della schiena, vincerà sugli altri imballi, creando un unico collo di 6 pezzi, in questo caso dobbiamo avere l'accortezza di mettere il numero max a 999.

E possibile dichiarare come montato anche il codice di vendita (macro), in questo caso non sarà necessario dichiarare il numero max a 999.

FM a Parte

Viene considerato per un imballo omogeneo, ad esempio se ho un imballo che prevede tre ripiani, se uno di questi è in fuori misura, viene considerato come imballo a se stante.

Forzatura Collo

0 no forza: non è impostata nessuna forzatura.

1 forza collo: considera il componente in questione sempre come collo a parte, ad esempio se ho un montato (su componente) e voglio creare un imballo a parte ad esempio per i ripiani in vetro, allora questi dovranno avere il forza collo.

2 forza su componente macro: ha lo stesso scopo del forza collo, la differenza è che la forzatura, viene utilizzata, quando non è un altro componente a decidere il montato, ma viene assegnato direttamente un collo (sempre montato), direttamente al codice di vendita (macro).

Bilancia Colli

Supponiamo si dover imballare 6 ripiani, i ripiani hanno un imballo, che prevede 5 pezzi per collo, l'algoritmo prevederebbe, un imballo con 5 pezzi e un imballo con 1 pezzo, spuntando questo flag, l'algoritmo bilancia i componenti negli imballi, cioè creando 2 colli con 3 componenti ciascuno.

No Collo

Questo tipo di imballo non crea nessuna informazione in *BORDINE*.

Ordinamento

Si tratta dell'algoritmo di ordinamento dei colli.



Figura 7.8: Ordinamento dei colli.

+ : ordinamento crescente;

- : ordinamento decrescente;

C : ordina per codice;

L A P ordina per dimensione;

B : ordina per box, forza l'accorpamento solo ai componenti dello stesso box ;

2 : ordina per abbinamento, in questo caso l'algoritmo privilegia i componenti appartenenti allo stesso box, ad esempio: supponiamo di avere una base da 45 più una base da 90 più una base da 45, assegniamo alle ante l'imballo con 2 pezzi, l'algoritmo, così di default, accorperà l'anta della base da 45 un'anta della base da 90, se invece, forziamo l'ordinamento a 2, l'algoritmo accorperà le 2 ante della base da 90 e le 2 ante delle 2 basi da 45.

7.1.2 Altri dati

In questi campi andremo a relazionare gli imballi con altri imballi presenti o nel box o nella macro, o nella composizione.

Identifichiamo:

- Raggruppa etichette;
- Altri pezzi da inserire;
- Codice imballo: serve per accorpare altri imballi sullo stesso imballo, prevede che sia spuntata una delle roundbox per la saturazione, nella fattispecie distinguiamo:

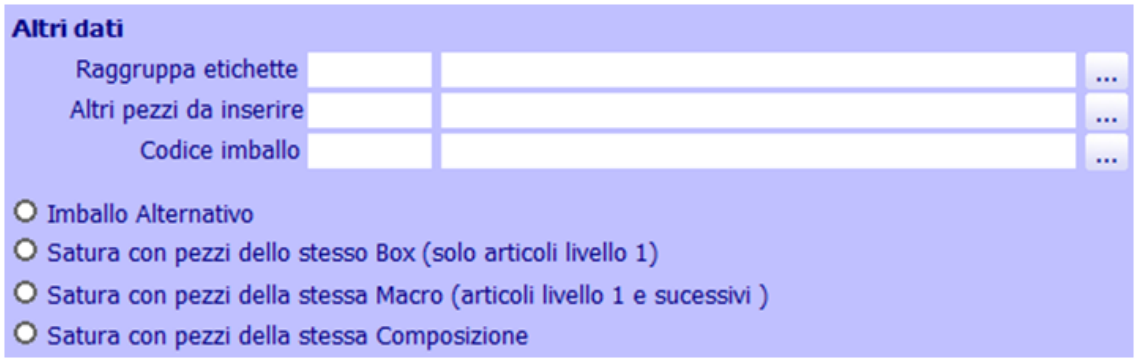

Altri dati
Raggruppa etichette
Altri pezzi da inserire
Codice imballo
Imballo Alternativo
Satura con pezzi dello stesso Box (solo articoli livello 1)
Satura con pezzi della stessa Macro (articoli livello 1 e sucessivi)
Satura con pezzi della stessa Composizione

Figura 7.9: Altri dati per imballi.

- Satura con pezzi dello stesso box: ovvero accorpa gli articoli/imballi appartenenti allo stesso box e di pari livello.
- Satura con pezzi della stessa macro: ovvero accorpa gli articoli/imballi appartenenti allo stesso box e livelli differenti.
- Satura con pezzi della stessa composizione: (da implementare).

7.2 Imballi associati

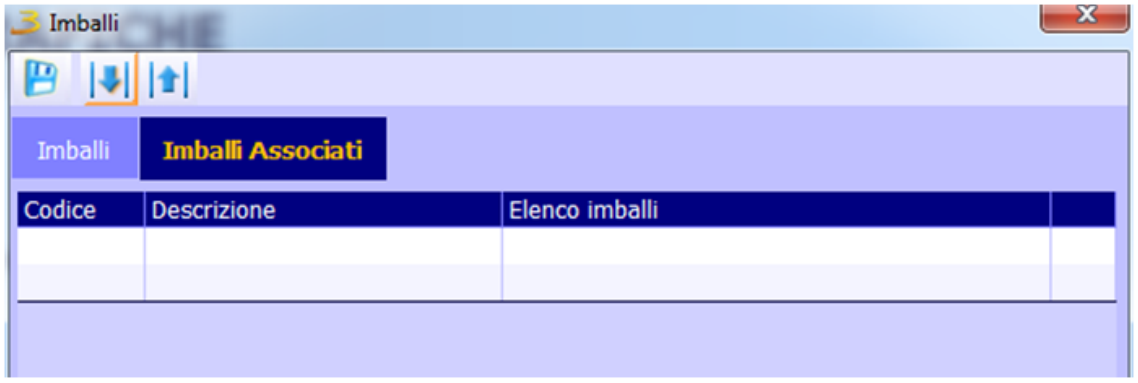


Figura 7.10: Imballi associati.

Da questa interfaccia possiamo associare 2 o più imballi in modo che possano convogliare in una unica bindella.

Vediamo di seguito un esempio di compilazione, Figura 7.11 nella pagina 261.

Quindi molto semplicemente ne deduciamo che:

1. Codice è una dichiarazione univoca, di tipo alfanumerico.
2. Descrizione è utile descrivere che cosa fanno i 2 (o più) imballi.
3. Elenco Imballi sono gli imballi che, se presenti in un ordine, vengono accorpati in un unico collo.

Imballi | **Imballi Associati**

Codice	Descrizione	Elenco imballi
1	SCHIENE18+VERGA	*SCH18*VERS*
2	box + attacaglie	*BOX*9XF*
3	ante scor.+formelle	*IMB2*FOR*
4	antecab + zoccolino	*ANTCAB*ANT*
5	schiene18+verghe	*IMV*VER*

Figura 7.11: Imballi associati, compilazione.

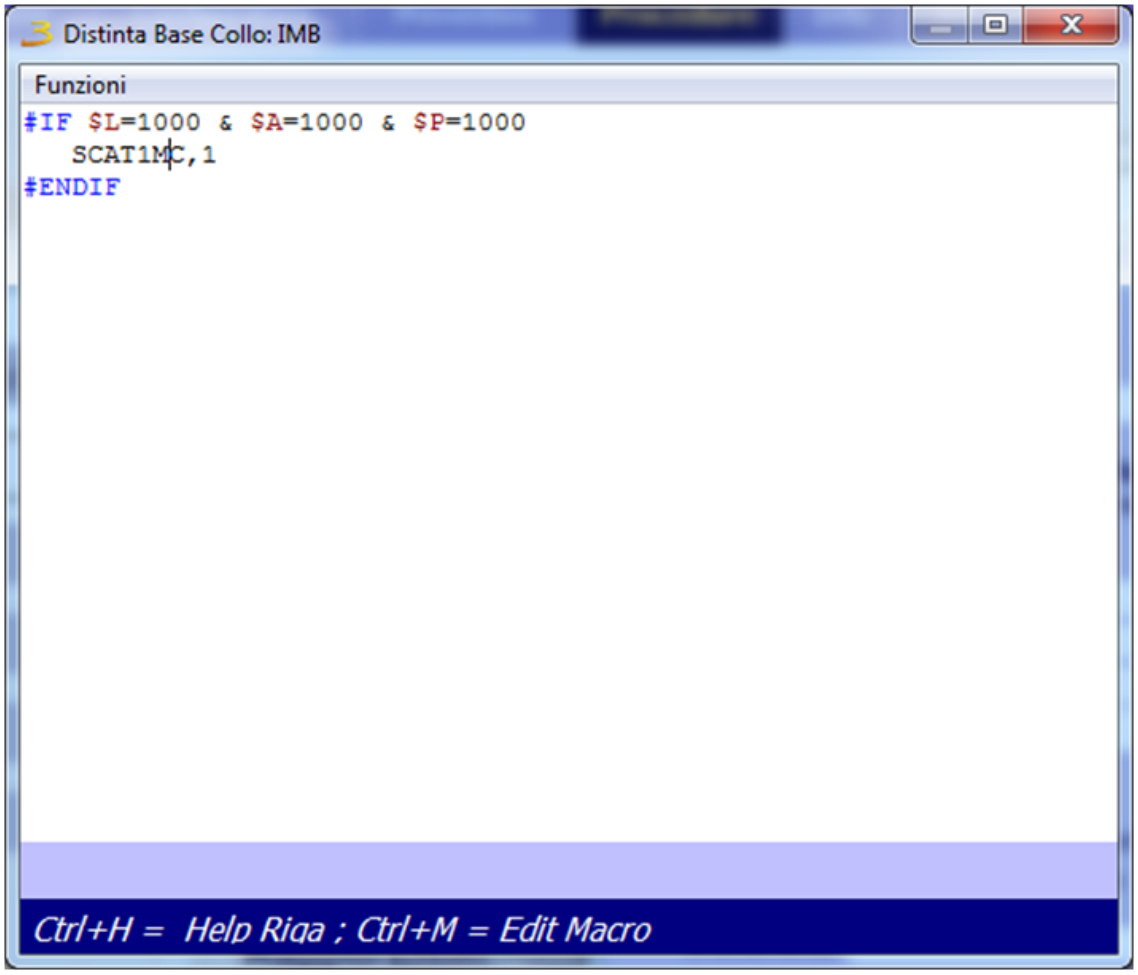


Figura 7.12: Distinta base collo.

Da questa interfaccia, Figura 7.12 nella pagina 261 possiamo, creare una vera e propria distinta a formule (come quella presente nelle tipologie), per gestire eventuali codici di, scatoloni, protezioni, angolari, polistiroli, ecc., cioè tutte quelle parti, che vanno a comporre l'imballo.

In questo ambiente abbiamo a disposizione le variabili $L, $A, $P, ovviamente riferite alle dimensioni dell'imballo.

I dati risultanti di questa distinta vengono salvati nella tabella *DISTINTAT* di ECADMASTER.

7.3 Dichiarazione in anagrafica articolo

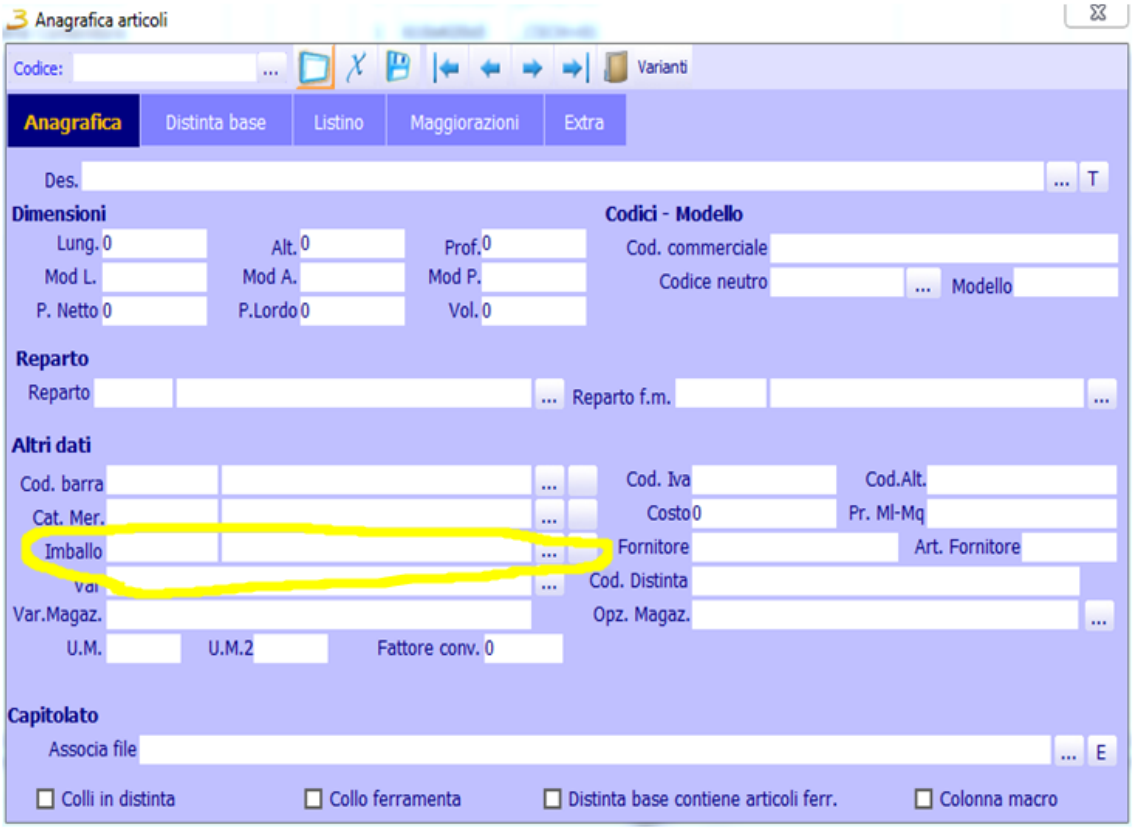


Figura 7.13: Anagrafica articoli, dichiarazione imballo.

Nel campo imballo dell'anagrafica articoli, dichiariamo nel primo campo il codice imballo o se preferiamo possiamo caricarlo facendo click su ..., il quale aprirà una finestra di ricerca dei codici imballo precedentemente creati.

Il secondo campo è dedicato alla descrizione caricata in automatico.

Da qui possiamo anche creare delle personalizzazioni o dei vincoli di raggruppamento, vediamo in che modo.

Supponiamo che il codice imballo sia IMB, che prevede di incorporare 6 componenti fianco, ricercati nello stesso ordine; nell'ordine esistono 3 contenitori però di diverse altezze, supponiamo uno alto 500mm uno alto 1500mm e l'ultimo alto 2500mm, a questo punto sorge la necessità che i fianchi vengano imballati a coppie con le stesse altezze.

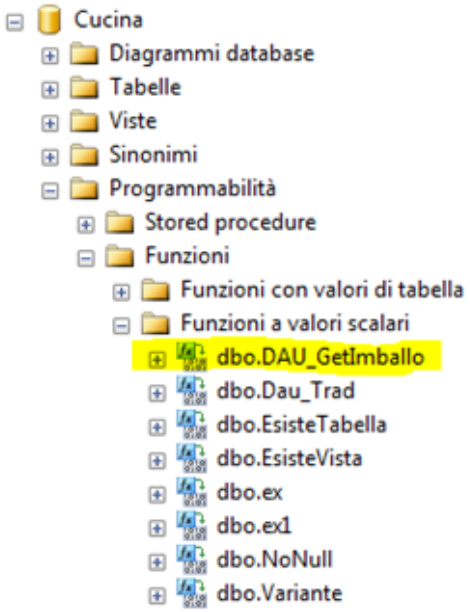
Possiamo quindi dichiarare nel campo imballo la seguente stringa:

IMB:L(R50)A(R50)

Andiamo ora ad analizzarla:

- IMB: è il codice imballo presente nell'anagrafica degli imballi (obbligatorio)
- [:] : è il separatore obbligatorio tra il codice imballo e gli altri dati;
- L(R50): significa che l'imballo incorporerà solo ed esclusivamente componenti che rientrano in un range di + 0 – 50mm in altezza l'uno dall'altro (facoltativo);
- A(R50): la stessa cosa solo che la dimensione di riferimento è l'altezza (facoltativo) Quindi essendo alcuni valori facoltativi, potremmo avere anche dichiarazioni del tipo: IMB, IMB:LA, IMB:L, IMB:L(R100)A, ecc.

7.4 Funzione `dau_getimballo`



La funzione `dau_getimballo`, che si trova nel database del catalogo sotto programmabilità->funzioni->funzioni con valori scalari, ci permette in fase di calcolo (salvataggio dell'ordine confermato) di cambiare l'algoritmo di riferimento (codice imballo).

Figura 7.14: Funzione `dau_getimballo`.

Facciamo un esempio:

Una coppia di fianchi sono imballati assieme, ma quando superano una certa misura devono essere imballati singolarmente, sapendo che nella funzione abbiamo i seguenti valori:

- Codice articolo;
- Dimensioni;
- Varianti;
- Codice imballo.

Possiamo così modificare la funzione, codice 216

Codice 216. Funzione `dau_getimballo`.

```
ALTER function [dbo].[DAU_GetImballo](
   @CODART varchar(20),
   @var varchar(512),
   @diml float,
   @dima float,
   @dimp float,
   @IMBALLO varchar(50)) returns varchar(50) as begin

   declare @cc varchar(50)
   set @cc = @IMBALLO

if LEFT(@cc,4) = 'FIA2' begin
  if @diml > 2200 and @dima > 560
    set @cc = 'IMB1' + substring(@cc,5,15)
end

  return @cc
end
```

8 Costruzione delle finiture

Nell'ambiente ***3CadEvolution*** , si chiamano finiture i frontali (ante e cassetti), i piani speciali (lavelli, piani cottura, top sagomati), le maniglie o pomoli, eventuali altri oggetti che per la loro forma necessitino di una definizione grafica più complessa di quanto si possa realizzare con una regola.

Per disegnare le ante è sufficiente far riferimento ad alcune funzioni specifiche, mentre nel caso di gestioni più complesse si rende necessaria la conoscenza di alcune nozioni di Visual Basic, per tanto, consigliamo al lettore di consultare la documentazione ad esso relativa, disponibile nel sito Microsoft.

Per accedere all'interfaccia di gestione delle finiture, dalla schermata dei materiali, selezionare il pulsante Script, si aprirà la finestra Gestione delle ante con il codice in VbScript. Questo file viene associato alla regola tramite inserimento della variabile *FACC* all'interno della regola con il valore dell'anta.

Esempio 1. Inserimento anta all'interno della regola.

FACC=22

Per tutti i comandi che seguono, il lettore ricordi che la posizione e le dimensioni del disegno si ottengono dalle coordinate iniziali e finali, e le eventuali misure da definire vanno espresse in millimetri.

Un altro modo per inserire le ante è utilizzare l'editor Help della regola, Figura 8.1, al quale si accede tramite Ctrl+H essendo posizionati sull'editor della grafica.

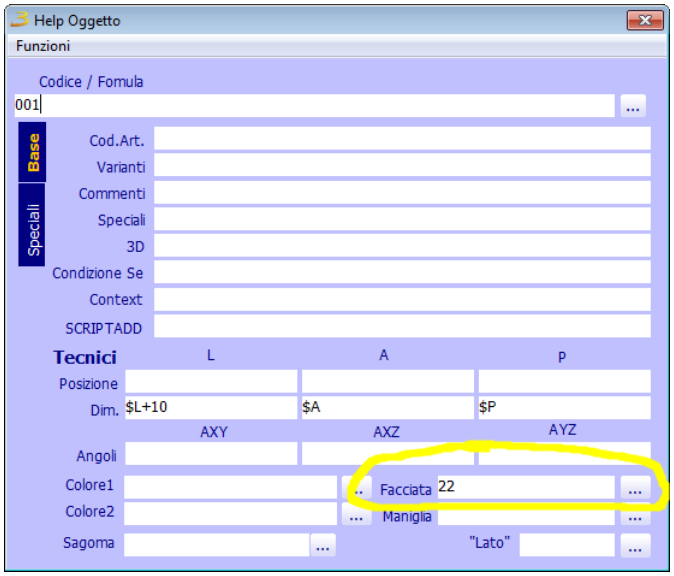


Figura 8.1: Editor help per inserire lo script delle ante.

Inserendo il numero dell'anta all'interno facciata viene creata in automatico la variabile *FACC*. Cliccando sul bottone ... si accede alla schermata Gestione delle ante.

8.1 Funzioni per la definizione delle finiture

8.1.1 Funzione `wrFaccia`

wrFaccia:

disegna solo la faccia di un cubo. In base al lato disegna la faccia di un determinato colore o senza linee bordo.

La sintassi da usare è la seguente:

wrFaccia (lato, dx, dy, dz, colore, lineeBordo)

- lato:
  - 0 : davanti
  - 1 : sopra
  - 2 : destra o sinistra;
- dx, dy, dz: dimensione in X, dimensione in Y, dimensione in Z;
- colore: colore della facciata;
- lineeBordo: linee bordo può assumere i valori true o false in base alla necessità di visualizzazione dei bordi o no.

8.1.2 Funzione `wrFacciaAlfa`

wrFacciaAlfa:

disegna una faccia trasparente.

La sintassi da usare è la seguente:

wrFaccia(dIx, dIy, dIz, colore, tipo, colore2)

dIx, dIy, dIz: imensioni

colore: materiale contenente immagine trasparente

tipo:

0 - faccia posizionata in 3D

1 -faccia che guarda sempre verso la camera

colore2

8.1.3 Funzione `wrScala`

wrScala:

aggiunge una scalatura a tutti gli oggetti dichiarati dopo la chiamata alla funzione. E' obbligatorio chiudere la scalatura con wrChiudi.

La sintassi da usare è la seguente:

```
wrScala(dx, dy, dz)
```

dx, dy, dz: fattori di scalatura.

8.1.4 Funzione `wrParametro`

wrParametro:

restituisce l'iesimo parametro della tipologia.

La sintassi da usare è la seguente:

```
wrParametro(i)
```

8.1.5 Funzione `wrTexAntaCurva`

wrTexAntaCurva:

La sintassi da usare è la seguente:

```
wrTexAntaCurva(sagoma, altezza, colore tex, direzione, rgb, bumpamp, XBUMP)
```

sagoma: punti della sagoma da disegnare;

altezza;

spessore;

colore;

tex: *Texture*;

direzione;

rgb: colore espresso in *RGB*;

bumpamp: ampiezza bump;

XBUMP: file della *Texture* bump.

8.1.6 Funzione `wrPanExt`

wrPanExt:

di solito tale funzione è in testa a tutte le finiture. La variabile modo serve da qualificatore per le funzioni Z, dove disegnare la facciata. Prima tale variabile aveva solo valori 1 e 2 (fronte e sopra), ora le possibilità sono le seguenti:

La sintassi da usare è la seguente:

wrPanExt(dx, dy, dz, c1, c2, modo)

- modo:
  - 1 disegna fronte
  - 2 disegna sopra
  - 3 disegna lato dx
  - 4 disegna fronte no schema
  - 5 disegna sopra no schema
  - 6 disegna lato dx no schema
  - 7 disegna dietro
  - 8 disegna sotto
  - 9 disegna lato sx
  - 10 disegna dietro no schema
  - 11 disegna sotto no schema
  - 12 disegna lato sx no schema
  - 13 disegna solo la rete del box;
- dx, dy, dz: dimnesioni in X,Y,Z;
- c1, c2: colore uno e colore due.

8.1.7 Funzione `wrPannello`

wrPannello:

disegna un pannello al quale si deve specificare un orientamento, in quanto in questa versione si potrà vedere anche la venatura del legno.

La sintassi da usare è la seguente:

wrPannello(orientamento, colore, dIx, dIy, dIz, s1, s2, s3)

orientamento: orientamento del sensovena;

colore;

dIx, dIy, dIz: dimensioni in X, Y, Z;

s1, s2, s3: punti raggio stondatura .

8.1.8 Funzione `wrPannello2`

wrPannello2:

questa funzione permette di disegnare un pannello (ad esempio un cubo) al quale si possono colorare tutti i lati in modo differente. L'ultimo parametro consente di tracciare il contorno se impostato a uno, altrimenti no se lasciato a zero.

La sintassi da usare è la seguente:

wrPannello2(dIx, dIy, dIz, xCDAVANTI, xCDIETRO, xCDESTRA, xCSINISTRA, xCSOPRA, xCSOTTO, faccia)

dIx,dIy,dIz: dimensioni in X, Y, Z;

xCDAVANTI: colore davanti;

xCDIETRO: colore dietro;

xCDESTRA: colore a destra;

xCSINISTRA: colore a sinistra;

xCSOPRA: colore sopra;

xCSOTTO: colore sotto;

faccia: linee bordo (true o false).

Esempio 2. Utilizzo della funzione `wrPannello2`.

Codice 217. Funzione `wrPannello2`.

```
1 nome3DS="catalogo\3ds\accessori\foto.jpg"
2 q1=wColore(163,-1,nome3ds)
3 wrPannello2 (dl),(da),(dp), q1, q1, q1, q1, q1, q1, 0
```

8.1.9 Funzione `wrDecalco`

wrDecalco:

aggiunge un decalco che viene applicata sui pannelli della scena secondo i parametri .

La sintassi da usare è la seguente:

wrDecalco(dx, dy, toll, ripetibile, colore, tipologie, suBox, spostaX, spostaY, ripX, ripY, inverti, extmaxx, extmaxy)

- dx, dy: dimensione decalco immaginandolo come singola faccia posizionata nello spazio;
- toll: offset che determina la distanza del decalco rispetto al pannello;
- ripetibile: determina se il decalco è ripetibile;
- colore: colore immagine decalco;
- tipologie: le tipologie su cui viene applicato il decalco;
- suBox: determina se il decalco deve essere applicato solo al box a cui appartiene;
- spostaX, spostaY: distanza tra una ripetizione e l'altra del decalco;
- ripX, ripY: numero delle ripetizioni nelle coordinate x e y;
- inverti: inverte la trasparenza originale dell'immagine;
- extmaxx, extmaxy: estensione massima di applicazione del decalco.

8.1.10 Funzione `wrTrasforma`

wrTrasforma:

stesso principio del wrScala con in più la possibilità di ruotare e traslare l'oggetto .

La sintassi da usare è la seguente:

wrTrasforma(px, py, pz, rAXZ, rAYZ, rAXY)

- px, py, pz: posizioni in X, Y, Z;
- rAXZ: rotazione AXZ;
- rAYZ: rotazione AYZ;
- rAXY: rotazione AXY.

8.1.11 Funzione `wrTrasformaPannello`

wrTrasformaPannello:

La sintassi da usare è la seguente:

wrTrasformaPannello(posizione in X, posizione in Y, posizione in Z, [rotazione AXZ], [rotazione AYZ], [rotazione AXY], [fattore scala X], [fattore scala Y], [fattore scala Z])

8.1.12 Funzione `wrChiudi`

wrChiudi:

chiude wrTrasforma.

La sintassi da usare è la seguente:

```
wrChiudi()
```

8.1.13 Funzione `wrSfera`

wrSfera:

disegna una sfera secondo i parametri specificati.

La sintassi da usare è la seguente:

```
wrSfera(dx, dy, dz, colore)
```

- dx, dy, dz: posizione in X, posizione in Y, posizione in Z.

8.1.14 Funzione `wrBox`

wrBox:

crea un box con le dimensioni e il colore specificati nei parametri.

La sintassi da usare è la seguente:

```
wrBox(px, py, pz, dx, dy, dz, colore)
```

- px, py, pz: posizioni in X, Y, Z;
- dx, dy, dz: dimensioni in X, Y, Z;
- colore: colore del box.

8.1.15 Funzione `wr3D`

wr3D:

carica un file *3DS*. I colori vengono sostituiti in base all'ordine degli oggetti presenti sul file 3D.

La sintassi da usare è la seguente:

wr3D(file, colore, dx, dy, dz, colore2, colore3 , colore4, colore5)

file: nome file;

colore: colore;

dx: dimensione in X;

dy: dimensione in Y;

dz: dimensione in Z;

colore2: secondo colore;

colore3: terzo colore;

colore4: quarto colore;

colore5: quinto colore.

8.1.16 Funzione `wr3D2`

wr3D2:

carica un file *3DS*. I colori vengono sostituiti in base all'ordine degli oggetti presenti sul file 3D. [9]

La sintassi da usare è la seguente:

Wr3d2(file, colore, dx,dy, dz, inix, iniy, iniz, finx, finy, finz, forzaUV)

file: nome file;

colore: colore;

dx, dy, dz: dimensione in X, Y, Z;

inix: determina la coordinata di inizio stretch in x partendo da sx dell'oggetto;

iniy: determina la coordinata di inizio stretch in y partendo dal basso dell'oggetto;

[9] Funzione obsoleta da sostituire con WRX2.

iniz: determina la coordinata di inizio stretch in z partendo dal retro dell'oggetto;

finx: determina la coordinata di inizio stretch in x partendo da dx dell'oggetto;

finy: determina la coordinata di inizio stretch in y partendo dall'alto dell'oggetto;

finz: determina la coordinata di inizio stretch in z partendo dalla fronte dell'oggetto;

forzaUV: forza ricalcolo coordinata della mappatura.

8.1.17 Funzione `wEstrudi`

wEstrudi:

estrude la sagoma.

La sintassi da usare è la seguente:

wEstrudi (tipo, sagoma, altezza, c1, c2, angolo)

tipo: tipo estrusione (vedi tabella);

sagoma: sagoma da estrudere;

altezza: altezza estrusione;

c1, c2: colore1 e colore2;

angolo: angolo inclinazione estruso.

8.1.18 Funzione `wrFacciaAlfa`

wrDoppioEstrudi:

comando utilizzato nelle finiture che consente di estrudere un oggetto con due sagome. Gestisce estrusioni di cornici con cambio altezza costante: corrimani e eliche.

La sintassi da usare è la seguente:

wrDoppioEstrudi(tipo, direttrice, sezione, c1, modo, angoloini, angolof, rapporto, altezza)

- tipo: (verso estrusione - 1);
  1 percorso profilo in pianta
  2 percorso profilo dall'alto in basso in profondità



Figura 8.2: Estrusione elica.

3 percorso profilo dall'alto in basso in fronte.

- direttrice: sagoma di direzione;
- sezione: sagoma della sezione;
- c1: colore;
- modo: obsoleto;
- angoloini, angolof: per sagome aperte angolo iniziale e finale;
- rapporto: coefficiente per calcolo mapping;
- altezza: differenza di altezza tra inizio e fine;

Esempio 3. Estrusione elica con il linguaggio delle regole.

```
1 !SAGOMA tmp C20, 0, 0, 30, 180
2 !SAGOMA tm2 C40, 300, 300, 300, 720
3 522, sa=0, l=$l, a=$a, p=$p, x2=1/0/500, dis=$(tmp), x1=$(tm2
    )
```

N.B: Il comando sagoma con cerchio:

!SAGOMA tmp C[numero punti], [x], [y], [raggio], [angolo iniziale], [angolo finale]

x2=1/0/500: è il parametro due della tipologia, dove 500 è l'altezza da raggiungere con l'elica.

Vecchia gestione: primo parametro = tipo(-1,1,2,3 indica le direzioni di estrusione) secondo = sagoma dell'anta sag1="#0,100,A10,0,100... terzo = sagoma del profilo sag2="#0,0,40,0,20,1000,0,1000" c1 = colore1 c2 = colore2

8.1.19 Funzione `wrTexAnta`

wrTexAnta:

per il disegno delle ante.

La sintassi da usare è la seguente:

---

wrTexAnta(dx, dy, dz, tex, incl, tx, ty, p1x, p1y, p2x, p2y, cx, cy, trasp, colore, rgb, bumpamp, XBUMP, texBW)

Parametri:

- dx = DL: dimensioni dell'anta in larghezza;
- dy = DA: dimensioni dell'anta in altezza;
- dz = DP: dimensioni dell'anta in profondità;
- tex: file *Texture* che rappresenta l'anta (.JPG);
- incl: fattore di piegatura dell'anta (tipicamente valori da 0 a metà dello spessore, provoca un effetto tronco di piramide);
- tx , ty = dx, dy: le dimensioni in mm cui è riferita la foto dell'anta (parametro tex);
- p1x, p1y, p2x, p2y: coordinate dei 2 punti (uno in basso a sinistra e uno in alto a destra) che delimitano il perimetro al di fuori del quale l'anta non va dimensionata (cornice dell'anta);
- cx: è un parametro opzionale, area che va tenuta ferma al centro del disegno e che non va ridimensionata (ad es. se c'è fiore da disegnare al centro dell'anta questo se allargo l'anta non si deve stirare). Se cx vale 1 l'anta risulta bucata. Valore di default è 0;
- cy: vedi cx;
- trasp: opzionale, percentuale di trasparenza che va da 0 a 1;
- colore: materiale base;
- rgb: colore;
- XBUMP: file della *Texture* bump;
- texBW: opzionale, contiene il percorso all'immagine da specificare nella stampa in bianco e nero.

8.1.20 Funzione `wrSpeciale`

wrSpeciale:

permette l'inserimento di una superficie di tipo speciale.

La sintassi da usare è la seguente:

wrSpeciale(tipo, ox, oy,dx, dy, speed, mat)

tipo=1 : flare ossia *Bitmap* additiva 3D

tipo=2 : flare 2D sul video (coordinate di riferimento -1,-1,1,1)

tipo=3 : proiezione diapositive

...

L'elenco dei materiali va separato con ; .

8.1.21 Funzione `wrCilindro`

wrCilindro:

La sintassi da usare è la seguente:

wrCilindro(raggio1, raggio2, lunghezza, spicchi, stacks, colore)

- raggio1;
- raggio2;
- lunghezza;
- spicchi;
- stacks;
- colore.

8.1.22 Funzione `WrToro`

WrToro:

disegna un toride.

La sintassi da usare è la seguente:

WrToro(innr, outer, rings, sides, idColore)

innr: raggio interno;

outr: raggio esterno;

rings: anelli;

sides: lati;

idColore: id colore.

8.1.23 Funzione `wColore`

wColore:

aggiunge materiale alla tabella dei materiali e ritorna l'indice del nuovo colore. comando per la gestione delle *Bump map*.

La sintassi da usare è la seguente:

---

wColore(nc, colore, tex, s1, s2, tradp, BItmapBW, effetto, xAng, Autobump, xBump, AMpBump)

- nc: materiale base riferimento;
- colore: colore;
- tex: nome file *Texture*;
- s1, s2: scalature in x,y della *Texture*;
- tradp: attiva/disattiva trasparenza;
- BitmapBW: nome file *Texture* sostitutiva stampe in bn;
- effetto: non usata, tenuta per compatibilità;
- xAng: angolo rotazione texture;
- Autobump: attiva bump automatico (simula rugosità superficie);
- xBump: file della *Texture* bump;
- AmpBum: ampiezza bump se è automatica.

Esempio 4. Creazione di una *Bump map*.

```
NEWIDCOLORE=wColore(
IDCOLORE, // rif. colore nella tabella colori
RGB, // -1 =default
TEXTURE // . =default
S1,S2, // float scala texture,
TRASPARENZA, // -1 =default (valori da 0 a 1)
FLAGBW, // disegna la stessa texture anche in BW se =1 (default 0)
EFFETTO, // non usato
ANGOLOROT, // default 0
AUTOBUMP, // se 1 genera autobump con ampiezza dichiarata
BUMP, // nome file specifico di bumpmap (autobump deve essere a 0)
 AMPBUMP) // ampiezza della bump generata (autobump=1) \\ \hline
WRDRAWTEXT: dx,dy,dz,stringa,tipo(0,1,2,3,4,5,6,7). Serve per disegnare una targhetta. \\ \hline
WRFACCIAALFA: dx,dy,dz,colore,tipo,colore2 \\
```

```
0=fronte
1=terra
2=profilo
3=stella
4=sandwitch
13=luce
```

8.1.24 Funzione `wrLibreria`

wrLibreria:

disegna una libreria. Questo comando richiede il catalogo "PARETI" installato.

La sintassi da usare è la seguente:

wrLibreria(dl, da, dp, tipo, percorsoLibri)

dl, da, dp: dimensioni libreria;

tipo: valori da 0 a 5;

percorsoLibri: percorso immagini con le copertine.

8.1.25 Funzione `wrLinea`

wrLinea:

disegna una linea su coordinate.

La sintassi da usare è la seguente:

wrLinea(x0, y0, z0, x1, y1, z1, colore, spessore)

x0, y0, z0: punto iniziale;

x1, y1, z1: punto finale;

colore: colore della linea;

pessore: spessore linea.

8.1.26 Funzione `wrLinee`

wrLinee:

disegna un vettore di linee.

La sintassi da usare è la seguente:

wrLinee(np, p(), modo, colore, spessore)

np: numero punti;

p(): vettore di punti;

modo:

0 -linee staccate

1 -percorso non chiuso

2 -percorso chiuso;

colore: colore delle linee;

spessore: spessore linea.

8.1.27 Funzione `wrMuro`

wrMuro:

questo comando fa parte unicamente del catalogo "PARETI".

La sintassi da usare è la seguente:

wrMuro(sagoma frontale, c1, c2, c3, spessore, corda, lunghezza, ang1, ang2)

8.1.28 Funzione `wrRivoluzione`

wrRivoluzione:

genera un solido di rivoluzione a partire da una sagoma.

La sintassi da usare è la seguente:

wrRivoluzione(px, py, pz, tipo, sagoma, c1, spicchi)

px, py, pz: centro della rivoluzione;

tipo:

1 - rivoluzione sull'asse X

2 - rivoluzione rispetto all'asse Y

3 - rivoluzione rispetto all'asse Z;

sagoma: punti del profilo;

c1: colore;

spicchi: numero spicchi;

8.1.29 Funzione `wrIniSagoma`

wrIniSagoma:

comando che ritorna i due punti iniziali x e y sono i parametri di ritorno"PARETI".

La sintassi da usare è la seguente:

```
wrIniSagoma(sagoma, x ,y)
```

8.1.30 Funzione `wrYSagoma`

wrYSagoma:

comando che torna il valore y della sagoma.

La sintassi da usare è la seguente:

```
wrYSagoma(sagoma, x)
```

8.1.31 Funzione `wrSimbolo`

wrSimbolo:

va utilizzata nella finitura della parete per aggiungere i simboli degli impianti all'interno dell'ambiente 3D.

La sintassi da usare è la seguente:

```
wrSimbolo(file, dimx, dimy, layer)
```

- file: è il nome del file immagine del simbolo;
- dimx, dimy: sono le dimensioni dell'oggetto;
- layer: è un intero che indica in modo univoco il layer dell'oggetto; si consiglia di utilizzare valori piccoli a partire da 1.

Attivazione in grafica I simboli compaiono selezionando Enable Layers all'interno del pannello viste.

> ! Cose da evitare. wrSimbolo si occupa già di disegnare il simbolo nella posizione corretta della parete. Il simbolo sarà centrato rispetto al punto di applicazione e assumerà dimensioni dimx, dimy. Evitare di aggiungere pannelli davanti al simbolo con l'immagine del simbolo stesso. Se il disegno del simbolo non compare a video in effetti è semplicemente perché il check Enable Layers è disabilitato all'interno di viste. Se nelle finiture che visualizzano il simbolo viene aggiunto un pannello posizionato erroneamente il risultato porta a delle quotature sbagliate).

8.1.32 Funzione `wrText`

wrText:

crea una quota con dei parametri configurabili. E' possibile creare una quota con o senza testo oppure solo un testo alternativo che rappresenta la quota stessa.

La sintassi da usare è la seguente:

wrText(tipo, testo, x1, y1, z1, x2, y2, z2, scala, distanza)

- tipo: tipo di quotatura, rappresentato con una stringa di caratteri, può assumere i seguenti valori:
  - q: quota normale;
  - L: posiziona il testo a sinistra della quota;
  - C: posiziona il testo al centro della quota;
  - R: posiziona il testo a destra della quota;
- testo: il testo che si vuole inserire;
- x1, y1, z1 e x2, y2, z2: i punti per i quali passa la quota;
- scala: la dimensione del testo;
- distanza: la distanza della quota dal disegno.

8.1.33 Funzione `wrXMov`

wrXMov:

serve per caricare un fileX animato.

La sintassi da usare è la seguente:

wrXMov(file)

- file: fileX da caricare.

8.1.34 Lista completa delle funzioni disponibili

in seguito è presente una lista completa delle funzioni disponibile.

Funzioni per le finiture
w2LineH(ByVal x0!, ByVal y0!, ByVal X1!, ByVal Y1!, ByVal colore, ByVal Spessore)
w2LineO(ByVal x0!, ByVal y0!, ByVal X1!, ByVal Y1!, ByVal colore, ByVal Spessore)
w2RectH(ByVal x0!, ByVal y0!, ByVal X1!, ByVal Y1!, ByVal colore, ByVal Spessore)
w2RectO(ByVal x0!, ByVal y0!, ByVal X1!, ByVal Y1!, ByVal colore, ByVal Spessore)
wChiudiNodo()
wColoreE(ByVal a$, Optional ByVal RuotaTex! = 0) As Long
wCreaNodo(ByVal idNodoStart) As Long
wDisegnaPiastrelle(ByVal idPadre, ByVal nn As Long, ByRef floatData() As Single, ByRef intData() As Long)
wEstrudi(ByVal tipo, ByVal Sagoma$, ByVal altezza!, ByVal c1&, ByVal c2&, Optional ByVal angolo! = 0, Optional ByVal SagomaArchivio = False, Optional ByVal altezzaBase! = 0)

Tabella 8.1 : continua nella prossima pagina

Tabella 8.1 : continua dalla pagina precedente

Funzioni per le finiture
wLaserSoff(ByVal idLista As Long, ByVal X As Single, ByVal z As Single, ByRef distance As Single)
wlBox(ByVal px!, ByVal py!, ByVal pz!, ByVal dx!, ByVal dy!, ByVal dz!, Optional ByVal colore&) As String
Wr3DSTexAnta(ByVal File3DS$, ByVal dx!, ByVal dy!, ByVal dz!, ByVal p1x!, ByVal p1y!, ByVal p2x!, ByVal p2y!, ByVal Tex$, ByVal tx!, ByVal ty!, Optional ByVal ori$ = XZY, Optional ByVal idColore = 200)
wrAbilitaluce(ByVal Luce, ByVal px!, ByVal py!, ByVal pz!, ByVal ox!, ByVal OY!, ByVal OZ!, Optional ByVal cc& = &HFFFFFF)
wrAddPuntoRighello(ByVal l As Long, ByVal a As Long, ByVal p As Long, ByVal codice As Long) As Long
wrAdvWebcamTexture()
wrAggiungiColoreEsteso(ByVal idbase As Long, ByVal ambient As Long, ByVal trasp!, ByVal texture$, ByVal scalax!, ByVal scalay!, ByVal rot!, ByVal bump$, ByVal TEXTUREbw$, ByVal metallica!, ByVal bumpamp!, ByVal refAmp!, ByVal autobump, ByVal shader) As Long
wrCompress3D(ByVal orig$, ByVal Dest$, ByVal forzamat As Long)
wrDragger(ByVal tipo, ByVal dm!, ByVal snap!, ByVal valori$, ByVal accettafm, ByVal Des$, ByVal colore, ByVal fscala!, Optional ByVal minimo!, Optional ByVal massimo!, Optional ByVal collidiBox As Integer = 0, Optional ByVal tipologie As String = ‘‘)
wrDrawText(ByVal dx!, ByVal dy!, ByVal dz!, ByVal stringa$, Optional ByVal tipo = 0, Optional ByVal tipo2 = 0, Optional ByVal dx2! = 0, Optional ByVal dy2! = 0)

Tabella 8.1 : continua nella prossima pagina

Tabella 8.1 : continua dalla pagina precedente

Funzioni per le finiture
wrDrawTextE(ByVal dx!, ByVal dy!, ByVal dz!, ByVal stringa$, ByVal tipo, ByVal tipo2, ByVal dx2!, ByVal dy2!, p1 As Punto3D, p2 As Punto3D, p3 As Punto3D)
wrFinSagoma(ByVal Sagoma$, X1 As Variant, Y1 As Variant)
wrLineBox(ByVal dx!, ByVal dy!, ByVal dz!)
wrLineeSagoma(ByVal punti As String, ByVal colore, ByVal Spessore!, ByVal Sagoma As Long, Optional ByVal aggiungiSAgoma As Long = 0)
wrLog(X As String)
wrLuce(ByVal tipo, ByVal tox, ByVal toy, ByVal toz, ByVal angolo1, ByVal angolo2, ByVal temperatura!)
wrMarker(ByVal dx As Single, ByVal dy As Single, ByVal dz As Single, ByVal posMarker As Long, ByVal desMarker As String, Optional ByVal idlayer As Long = -1) As Long
wrMuro(Sagoma$, ByVal Colore1, ByVal Colore2, ByVal colore3, ByVal Spessore, ByVal corda!, ByVal Lunghezza!, ByVal angolo1!, ByVal angolo2!)
WrNewAnta(ByVal Sagoma$, ByVal h!, ByVal Spessore!, ByVal angolo1!, ByVal angolo2!, ByVal colore!, Optional ByVal CDIETRO! = -1, Optional ByVal cLato! = -1, Optional ByVal CSOPRA = -1, Optional ByVal h1! = 0, Optional ByVal h2! = -1)
wrNewAntaAddUV(ByVal nodo, ByVal modo, ByVal u!, ByVal uValue!)

Tabella 8.1 : continua nella prossima pagina

Tabella 8.1 : continua dalla pagina precedente

Funzioni per le finiture
wrNodeAdd3(ByVal idPadre, ByVal livello, ByVal px!, ByVal py!, ByVal pz!, ByVal AXZ!, ByVal AYZ!, ByVal AXY!, ByVal scalax!, ByVal scalay!, ByVal scalaz!, ByVal ditta, ByVal tipologia, ByVal statotop As Long, ByVal mdx!, ByVal mdy!, ByVal mdz!, ByVal modoAngolo, ByVal Layer, ByVal tipoSimm, ByVal simmX!, ByVal simmY!, ByVal simmZ!) As Long
wrIniSagoma(ByVal Sagoma$, X1 As Variant, Y1 As Variant)
wrNodeAdd3DDau(ByVal idPadre, ByVal id3Ddau, ByVal nco, co) As Long
wrNodeSetModifier(ByVal id&, Optional ByVal r = 0, Optional ByVal g = 0, Optional ByVal B = 0, Optional ByVal alfa = 255, Optional ByVal Spessore As Single = 1)
wrNodoEstruso(ByVal lato As Long, ByVal hExtr!, ByVal posNeg As Long, ByVal idSagoma As Long, ByVal tipoMappa As Long, ByVal c1 As Long, Optional ByVal c2 As Long = -1, Optional ByVal c3 As Long = -1, Optional ByVal c4 As Long = -1, Optional ByVal c5 As Long = -1, Optional ByVal c6 As Long = -1, Optional ByVal orienta$ = , Optional ByVal colLav As Long = -1) As Long
wrOggetto2(ByVal Sagoma$)
wrPanExt(ByVal dx!, ByVal dy!, ByVal dz!, ByVal c1&, ByVal c2&, ByVal modo)
WrPannEffetto(ByVal idEffetto, ByVal dx!, ByVal dy!, ByVal dz!, ByVal NomeFile$, ByVal parametri$)
WrParticle(ByVal accx!, ByVal accY!, ByVal accZ!, ByVal maxPart, ByVal pTime!, ByVal systemType, ByVal idMat, ByVal size!)
wrPavimento()

Tabella 8.1 : continua nella prossima pagina

Tabella 8.1 : continua dalla pagina precedente

Funzioni per le finiture
wrSfera(ByVal dx!, ByVal dy!, ByVal dz!, ByVal cc&, Optional ByVal spicchi As Long = 16)
wrSparaLaserTipologie(ByVal x0!, ByVal y0!, ByVal z0!, ByVal X1!, ByVal Y1!, ByVal z1!, ByVal tipologie$, ByRef distance!, ByRef id2 As Long) As Long
WrTeapot(ByVal scalax!, ByVal scalay!, ByVal scalaz!, ByVal idColore)
wrTimeSeq(ByVal file$, Optional ByVal RIPETI = 0, Optional ByVal Famiglia = 0) As Long
wrTrasformaPannello(ByVal px!, ByVal py!, ByVal pz!, Optional ByVal AXZ! = 0, Optional ByVal AYZ! = 0, Optional ByVal AXY! = 0, Optional ByVal scalax! = 1, Optional ByVal scalay! = 1, Optional ByVal scalaz! = 1, Optional ByVal tiposim, Optional ByVal simx!, Optional ByVal simy!, Optional ByVal simz!) As Long
wrXcollision(ByVal file$, ByVal alfa, ByVal Colori$, Optional ByVal dx! = 0, Optional ByVal dy! = 0, Optional ByVal dz! = 0)
wScriviSagoma(ByVal s$) As Long

8.2 Utilizzo file .x

L'utilizzo dei file .x diventa necessario nell'istante in cui è richiesto un oggetto solido (tipo *3DS*) ma con una maggiore flessibilità dal punto di vista della scalabilità e colorazione.

Resta inteso che la creazione di un file .x resta una procedura complessa e riservata agli utilizzatori di programmi di manipolazione tridimensionale, quali ***3dstudiomax***, ***rinocerus***, ecc.

Si può prevedere con appositi programmi (tipo ***deep exploration***) anche una conversione da *3DS* a file .x.

8.2.1 Funzioni WRX e WRX2

Questi sono i 2 comandi per richiamare all'interno delle finiture il file di tipo *.x* .

I file *CAD* esterni sono gestiti da 3CAD nel formato *3DS* e nel formato X. La distinzione tra i due formati è non solo tecnica, ma sostanziale per la gestione della parametricità e dei colori: il file *3DS* viene convertito in OGG e la sua gestione dei colori è sempre e comunque parametrica (compatibile 3CAD Classic), passando per la funzione WR3D (file,colore,dl,da,dp,colore2...), il file X utilizza una nuova gestione più efficace, in quanto permette di definire i layers dei colori sostituibili (codice bitmap =01, 02... 20) e quindi consente un maggiore controllo sui dati. E' possibile richiamare anche un *3DS* con la funzione WRX e WRX2, rendendo di fatto la gestione del *3DS* e dei file X identica. Il vantaggio di utilizzare questa funzione è la possibilità che offre in termini di compressione dei dati (formato DR3D circa 1/3 del formato *3DS* con compressione attiva.) e della maggiore parametricità.

Le funzioni WRX e WRX2, permettono inoltre di accedere ai file di tipo script (estensione .XP) che funzionano come le finiture delle ante, ma sono portabili su ogni catalogo.

WRX:

La sintassi da usare è la seguente:

WRX(file : stringa obligatoria alfa : inutilizzato Colori : stringa facoltativa dx : numerico facoltativo dy : numerico facoltativo dz : numerico facoltativo modomat : numerico facoltativo pPivotL : numerico facoltatico, default -1 pPivotA : numerico facoltatico, default -1 pPivotP : numerico facoltatico, default -1)

- file: nome e percorso file;
- alfa: parametro inutilizzato al momento, mantenuto per questioni di compatibilità;
- colori: elenco colori separati da [,] ;
- dx: dimensione in larghezza;
- dy: dimensione in altezza;
- dz: dimensione in profondità;
- funzione: definisce il tipo di strecciabilità dell'oggetto, tabella 8.2;
- modomat: da implementare ora di default è 0. Determinerà se su quel oggetto andranno applicati i materiali dell'oggetto o se i materiali saranno specificati e sostituiti dal programma-,
- pPivotL: percentuale di spostamento dell'origine dell'oggetto rispetto all'asse delle x, in sostanza forza il pivot in X, reimpostando l'origine;

- pPivotA: percentuale di spostamento dell'origine dell'oggetto rispetto all'asse delle y, in sostanza forza il pivot in Y, reimpostando l'origine;
- pPivotP: percentuale di spostamento dell'origine dell'oggetto rispetto all'asse delle z, in sostanza forza il pivot in Z, reimpostando l'origine.

Tipi di strecciabilità

Valore	Descrizione
0	non applica le funzioni di stretching evolute ma si limita a scalare l'oggetto sino a portarlo alle dimensioni richieste (dimX/dimY/dimZ).
1	applica la scalatura estesa. L'oggetto viene scalato mantenendo però le dimensioni originali delle fasce esterne specificate coninix/finix/iniy/finiy/iniz/finiz (es figRef) L'oggetto viene anche rimappato secondo Larghezza/Altezza.
2	applica la scalatura estesa come in 1 ma mappando l'oggetto secondo Altezza/Larghezza.
101	applica la scalatura estesa come in 1 ma lasciando inalterata la mappatura UV.
300	applica la scalatura estesa fissando però la parte centrale dell'oggetto secondo le dimensioni. Specificate esclusivamente lungo l'asse X mentre le altre le streccia eventualmente sproporzionandole. L'effetto sarà quello di mantenere la parte originale del disegno uguale a quella originale (es Figura 8.4 nella pagina 289). Non altera la mappatura UV.
301	scalatura che mantiene fissa la parte centrale Y. Non altera la mappatura UV.
302	scalatura che mantiene fissa la parte centrale Z. Non altera la mappatura UV.
303	scalatura che mantiene fissa la parte centrale XY. Non altera la mappatura UV.
304	scalatura che mantiene fissa la parte centrale XZ. Non altera la mappatura UV.
305	scalatura che mantiene fissa la parte centrale YZ. Non altera la mappatura UV.
310	scalatura che mantiene fissa la parte centrale X. Applica nuova mappatura Altezza/Larghezza.
311	scalatura che mantiene fissa la parte centrale Y. Applica nuova mappatura Altezza/Larghezza.
312	scalatura che mantiene fissa la parte centrale Z. Applica nuova mappatura Altezza/Larghezza.
313	scalatura che mantiene fissa la parte centrale XY. Applica nuova mappatura Altezza/Larghezza.

Tabella 8.2 : continua nella prossima pagina

Tabella 8.2 : continua dalla pagina precedente

Valore	Descrizione
314	scalatura che mantiene fissa la parte centrale XZ. Applica nuova mappatura Altezza/Larghezza.
315	scalatura che mantiene fissa la parte centrale YZ. Applica nuova mappatura Altezza/Larghezza.

Tabella 8.2: Tipi di strecciabilità.

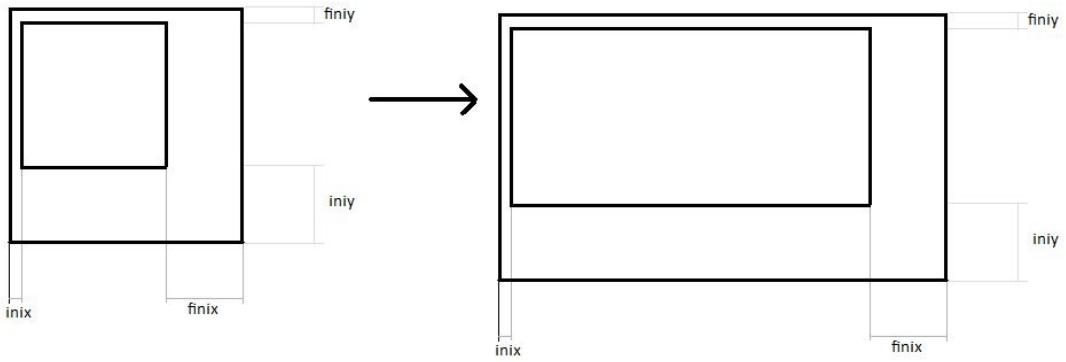


Figura 8.3: Esterno.

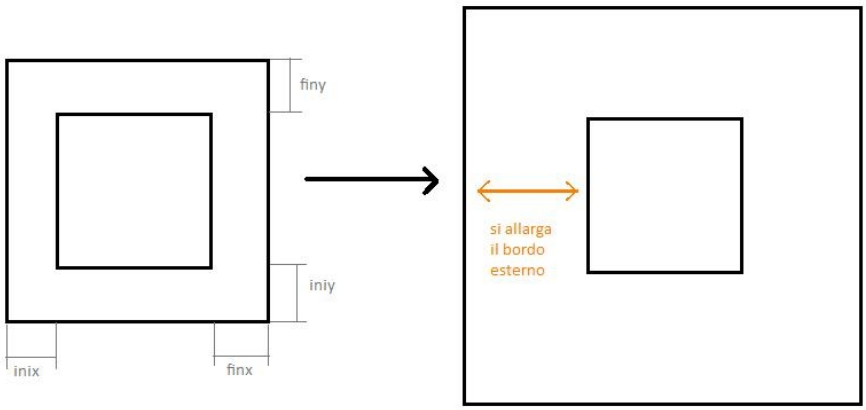


Figura 8.4: Interno.

Esempio 5. Utilizzo file .x .
Un'anta a telaio che necessita di essere ridimensionabile in larghezza e altezza ma non nella larghezza del telaio stesso.

Codice 218. Funzione `wrx2`.

```
Sub Anta10(dl, da, dp, c1, c2, pTerra, lato, px, py, pa,
    pAltezza, pLarghezza)
nome3d=ex(zParametro(6), "/", 1) //nome 3ds passato nel
    parametro 6
xcol1=ex(zParametro(6), "2") //colore passato nel parametro
    6, seconda posizione
xcol1=ex(zParametro(6), "3") //colore passato nel parametro
    6, terza posizione
File3D=percorso&"/3ds/"&nome3d //percorso e nome file
c1=wcolore(xcoll1) //assegnazione colore
c2=wcolore(xcol2) //assegnazione colore
Colori= c1 & "," c2
wrx2 file3d, 1, Colori, dl, da, dp, 3, "50;50;50;50;0;0"
end sub
```

9 Funzione elabora finale

L'elabora finale è una procedura che viene eseguita al momento del ricalcolo finale dei top, alla generazione del preventivo o al salvataggio dell'ordine.

Tale procedura va a testare il posizionamento di una tipologia grafica rispetto ad un'altra appartenente a un box diverso e permette di modificare le varianti di una regola (scelta sconsigliata) o i parametri di una tipologia sulla base del posizionamento relativo tra le stesse, determinando in particolare quando quest'ultime sono adiacenti.

Tale procedura è da utilizzarsi in particolare quando vi è la necessità di eseguire un ricalcolo finale sulle tipologie appartenenti a box diversi. Il caso tipo riguarda la gestione della laccatura dei fianchi sulla base dei box vicini, caso che sarà qui visto come esempio.

E' necessario innanzitutto generare un file .vbs (esempio job_fianchi.vbs) e salvarlo all'interno della cartella del catalogo.
Successivamente inserire nel file *ecadpro.ini*(nel gruppo [INFOGEN]) del catalogo la voce:

elaborafinale= elenco file per elabora finale, separati da [;].

Esempio 1. Flag elaborafinale.

elaborafinale=job_fianchi;job_zoccoli

Gli script di tipo job finale vengono richiamati in sequenza nell'ordine specificato in *ecadpro.ini*. Ogni script deve contenere 4 funzioni:

1. JobMain.
2. JobIni.
3. JobPannello.
4. JobFin.

JobMain viene richiamata una volta sola. Può effettuare quindi operazioni su tutto il progetto. Se ritorna una stringa di coppie di tipologie incise e incidenti:

T, TI1\nT, TI2\nT, TI3\n

Allora *per ogni coppia* vengono chiamate in sequenza le funzioni: JobIni, JobPannello e JobFin. JobIni, JobPannello, JobFin: se viene cambiata una variante o una proprietà di una tipologia (pannello) bisogna segnalarlo con la proprietà *fljob* del pannello.

- fljob=1: se viene cambiata una variante e quindi bisogna rivalutare tutta la regola;
- fljob=2: se viene cambiata una proprietà e quindi bisogna rivalutare solo la tipologia.

9.1 Public Function JobMain

In questa funzione va dichiarato l'elenco delle tipologie di cui si andrà a testare il posizionamento.

La sintassi da usare è la seguente:
Public Function JobMain(var)

```
Public Function JobMain(var)
Dim aa
aa= "501,502 // Tipologia fianchi su laterali \n"
aa= aa & "502,501 // Tipologia fianchi su laterali \n"
JobMain=replace(aa,"\n",vbcrlf)
End Function
```

Nel caso specifico abbiamo dichiarato che si vogliono testare le tipologie dei fianchi l'una contro l'altra, fianco sinistro contro fianco destro e viceversa.

N.B. Si può lavorare solo con tipologie padri non con le figlie (es:501.001...).

Variabili per elabora finale

Variabile	Descrizione
xamb.nbox	rilascia il numero totale di box presenti in grafica
x.DistanzaPavimento	rilascia la distanza dal pavimento del box
x.DistanzaBoxDavanti	rilascia la distanza tra il box corrente e il box davanti
x.DistanzaBoxDietro	rilascia la distanza tra il box corrente e il box dietro
x.DistanzaBoxSinistra	rilascia la distanza tra il box corrente e il box a sinistra
x.DistanzaBoxDestra	rilascia la distanza tra il box corrente e il box a destra
x.DistanzaBoxSopra	rilascia la distanza tra il box corrente e il box sopra
x.DistanzaBoxSotto	rilascia la distanza tra il box corrente e il box sotto
x.IdBoxDavanti()	rilascia l'Id (il numero di box) che è davanti al box corrente
x. IdBoxDietro()	rilascia l'Id (il numero di box) che è dietro al box corrente
x. IdBoxSinistra()	rilascia l'Id (il numero di box) che è a sinistra al box corrente
x. IdBoxDestra()	rilascia l'Id (il numero di box) che è a destra al box corrente
x. IdBoxSopra()	rilascia l'Id (il numero di box) che è sopra al box corrente
x. IdBoxSotto()	rilascia l'Id (il numero di box) che è sotto al box corrente se non c'è nessun box in corrispondenza della direzione che si sta testando la variabile rilascia -1

Tabella 9.1: Variabili per job finale.

Esempio 2. Script per modificare alcune varianti nella regola.

```
option explicit
dim xamb
public function jobmain(var)
dim i,a1,x
set xamb=amb
for i=0 to xamb.nbox-1
set x=xamb.box(i)
if x.regola<>"" then
if x.varRegola("QuotBase")<>"" then
a1=x.DistanzaPavimento
if a1<200 then
x.varRegola("QuotBase")="01"
else
x.varRegola("QuotBase")="02"
end if
end if
if x.varRegola("DelQuot")<>"" then
```

```
a1=x.idBoxSinistra()
if a1<0 then
x.varRegola("DelQuot")="01"
else
x.varRegola("DelQuot")="02"
end if
end if
end if
next
end function
```

9.2 Public sub JobIni

Lo scopo di tale funzione è principalmente quello di permettere di salvare i dati della tipologia principale (la prima dichiarata nella funzione JobMain, ossia nel primo passaggio 501, nel secondo 502) prima di procedere alla modifica degli stessi con le elaborazioni successive, per poter ripristinare correttamente la situazione iniziale ogni qualvolta si rilancia la procedura.

La sintassi da usare è la seguente:
Public sub JobIni(p, var)

Esempio 3. Utilizzo funzione JobIni.

```
Public Sub JobIni(p,var)
Dim fl
Dim tmp_tiplac,tmp_codlac,tmp,tmp_p2,tmp_p3
fl=0 'parametro che attiva il ricalcolo della tipologia
tmp_codlac=p.Parametro(2)
tmp_tiplac=p.Parametro(3)
If tmp_tiplac="" Then 'se parametro 2 e' vuoto, no ho ancora
    salvato nulla
tmp_tiplac=p.Parametro(7)
Else 'ripristino i parametro 7 iniziale che avevo memorizzato
     in parametro 3 della tipologia
If tmp_tiplac<>p.parametro(7) Then
  p.parametro(7)=tmp_tiplac:fl=1
End If
End If
If tmp_codlac="" Then 'se parametro 2 e' vuoto, no ho ancora
    salvato nulla
tmp_codlac=ex(p.Parametro(9),"/",3)
Else 'ripristino i parametro 9 iniziale che avevo memorizzato
     in parametro 2 della tipologia
If tmp_codlac<>ex(p.Parametro(9),"/",3) Then
  tmp=inseriscistringa(p.parametro(9),tmp_codlac,"/",3)
  p.parametro(9)=tmp:fl=1
End If
End If
tmp_p2=tmp_codlac
tmp_p3=tmp_tiplac
```

```
If p.parametro(2)<>tmp_p2 Then 'salvo parametro 9/3 originale
    in parametro 2
p.parametro(2)=tmp_p2:fl=1
End If
If p.parametro(3)<>tmp_p3 Then 'salvo parametro 7 originale
   in parametro 3
p.parametro(3)=tmp_p3:fl=1
End If
If fl Then p.flJob=1 'ricalcolo tipologia principale
End Sub
```

La variabile p permette di richiamare i parametri della tipologia principale e di andare a modificarla.

N.B: La variabile p.fljob attiva il ricalcalo e quindi ridisegna la tipologia, ogni qualvolta si modifica una tipologia è necessario impostare questa variabile a 1 per rendere effettiva la modifica.

9.3 Public Sub JobPannello

La procedura prevede di entrare in questa funzione solo e solo se le tipologie dichiarate sopra sono adiacenti.

La sintassi da usare è la seguente:
Public Sub JobPannello(p, p1, var, lati, pl, pa, pp, ax, ay, az)

Variabili della funzione JobPanello

Variabile	Descrizione
p	parametri della tipologia principale
p1	parametri della tipologia secondaria
var	?
lati	?
pl	posizione relativa in larghezza della tipologia secondaria rispetto alla principale
pa	posizione relativa in altezza della tipologia secondaria rispetto alla principale
pp	posizione relativa in profondità della tipologia secondaria rispetto alla principale
ax	rotazione relativa in x della tipologia secondaria rispetto alla principale
ay	rotazione relativa in y della tipologia secondaria rispetto alla principale
az	rotazione relativa in z della tipologia secondaria rispetto alla principale

Tabella 9.2 : continua nella prossima pagina

Tabella 9.2 : continua dalla pagina precedente

Variabile	Descrizione

Tabella 9.2: Variabili della funzione JobPannello.

Altre variabili:

p.xbox.xbox.pa : restituisce la posizione in altezza all'interno del disegno del box che contiene la tipologia principale.

Esempio 4. Utilizzo funzione JobPannello.

```
Public Sub JobPannello(p,p1,var,lati,pl,pa,pp,ax,ay,az)
Dim new_tiplac,new_codlac,tmp,p_pa,p1_pa,p_pp,p1_pp
'assegnazione variabili posizionamento
p_pa=p.xbox.xbox.pa-1*calcola(p.parametro(5))+p.pa
p1_pa=p1.xbox.xbox.pa-1*calcola(p1.parametro(5))+p1.pa
p_pp=p.xbox.xbox.pp+p.pp
p1_pp=p1.xbox.xbox.pp+p1.pp
'fianco destro
If p.tipo=502 And (p_pa>=p1_pa And (p.da+p_pa)<=(p1.da+
    p1_pa)) And (p_pp>=p1_pp And (p.dp+p_pp)<=(p1.dp+p1_pp)
    ) Then
  'allora il fianco e' completamente coperto dal fianco sx
      dell'altro box, quindi non va laccato
  new_tiplac=replace(ex(p.Parametro(7),"/",1) ,"FS","") '
     non va laccato fronte-sinistra
  tmp=inseriscistringa(p.parametro(7),new_tiplac,"/",1) '
     modifica parametro per finitura
  p.parametro(7)=tmp
  new_codlac = "FLF"
  tmp=inseriscistringa(p.parametro(9),new_codlac,"/",3) '
     modifica parametro per distinta base
  p.parametro(9)=tmp
  p.flJob=1 'ricalcola fianco
End If
'fianco sinistro
If p.tipo=501 And (p_pa>=p1_pa And (p.da+p_pa)<=(p1.da+
    p1_pa)) And (p_pp>=p1_pp And (p.dp+p_pp)<=(p1.dp+p1_pp)
    ) Then
  'allora il fianco e' completamente coperto dal fianco dx
      dell'altro box, quindi non va laccato
  new_tiplac=replace(ex(p.Parametro(7),"/",1),"FD","") '
     non va laccato fronte-sinistra
  tmp=inseriscistringa(p.parametro(7),new_tiplac,"/",1) '
     modifica parametro per finitura
  p.parametro(7)=tmp
  new_codlac = "FLF"
  tmp=inseriscistringa(p.parametro(9),new_codlac,"/",3) '
     modifica parametro per distinta base
  p.parametro(9)=tmp
  p.flJob=1 'ricalcola fianco
End If
```

9.4 Public sub JobFin

In questa funzione la procedura entra sempre dopo le altre funzioni e permette di modificare i parametri della tipologia principale.

La sintassi da usare è la seguente:
Public sub JobFin(p, var)

10 Print Server

10.1 Configurazione

Il servizio di stampa ***3CadEvolution*** permette l'esecuzione automatica di due procedure:

1. Stampa degli ordini.
2. Download ordini inviati tramite ***eCadLite*** .

Le due procedure possono essere eseguite, in base alla configurazione, in sequenza oppure separatamente.

Il programma è contenuto all'interno del file *PrintSrvPro.dll* e viene installato insieme al programma; il percorso di installazione è la cartella `_eCadPro\Procedure`.

Per avviare il PrintSrvPro è necessario avviare ***3CadEvolution*** con dei parametri aggiuntivi:

- Avvio servizio di stampa standard

```
C:\evolution\ecadpro\eCadPro.exe /C nomecatalogo /P
 E:\evolution\ecadpro,
 /START printsrvpro.crea
```

- Avvio servizio di stampa automatico

```
C:\evolution\ecadpro\eCadPro.exe /C nomecatalogo /P
 E:\evolution\ecadpro
 /START printsrvpro.crea, /auto
```

- Avvio servizio di download ordini

```
C:\evolution\ecadpro\eCadPro.exe /C nomecatalogo /P
 E:\evolution\ecadpro,
 /START printsrvpro.crea,auto /D
```

- Avvio servizio di download ordini automatico

```
C:\evolution\ecadpro\eCadPro.exe /C nomecatalogo /P
 E:\evolution\ecadpro,
 /START printsrvpro.crea, /D /AUTO
```

- Avvio servizio di controllo e avviso tramite email della scadenza licenze ecadlte

```
C:\evolution\ecadpro\eCadPro.exe /C nomecatalogo /P
 E:\evolution\ecadpro,
 /START printsrvpro.crea, /LIC
```

- Avvio servizio di controllo e avviso tramite email della scadenza licenze ecadlte automatico

```
C:\evolution\ecadpro\eCadPro.exe /C nomecatalogo /P
 E:\evolution\ecadpro,
 /START printsrvpro.crea, /LIC /AUTO
```

I parametri possono anche essere impostati in sequenza il modo che il Print Server esegua tutte le funzionalità in sequenza:

Esempio 1. Impostazione di tutte le funzionalità di Print Server contemporaneamente.

```
C:\evolution\ecadpro\eCadPro.exe /C nomecatalogo /P
E:\evolution\ecadpro /START printsrvpro.crea, /D /LIC /AUTO
```

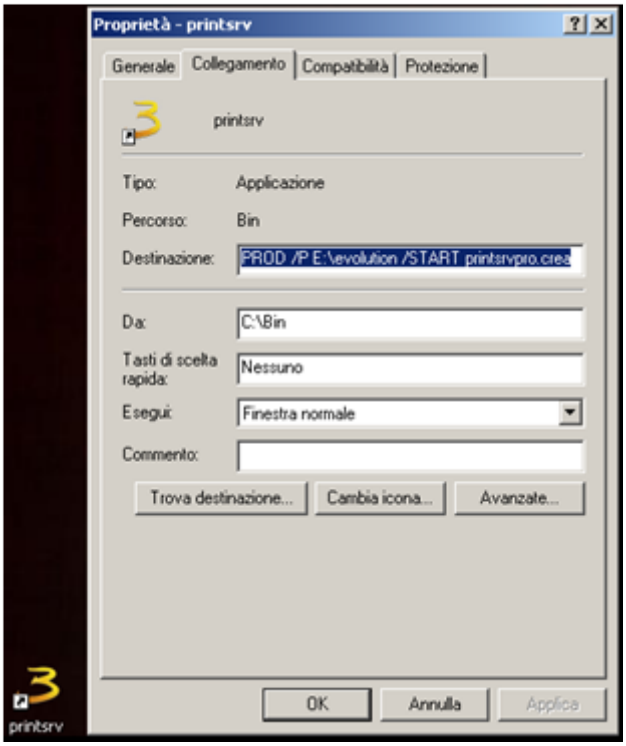


Figura 10.1: Impostazione Print Server.

Fanno parte del servizio Print Server di ***3CadEvolution*** anche il file di script *Ambiente.vbs*, il file *SalvaDatabase.vbs* e la tabella *PrintSrv* all'interno del database eCadMaster. I file di script sono necessari perché contengono tutte le personalizzazioni cliente. La tabella *PrintSrv* è necessaria perché è la base dati su cui il servizio si appoggia per sapere quali ordini deve stampare.

10.2 Stampa ordini

La stampa degli ordini viene eseguita dal printServer leggendo i dati inseriti all'interno della tabella printSrv, a sua volta popolata attraverso una personalizzazione della procedura di salvataggio ordini all'interno del file *SalvaDatabase.vbs*.

La procedura personalizzata stabilisce infatti quali ordini inseriti all'interno della tabella, specificando per ognuno di questi, il tipo di operazione che deve essere eseguita. Il campo Operazione della tabella può assumere diversi valori numerici, ognuno dei quali definisce come deve essere trattato l'ordine in quel momento processato.

I valori numerici che questo campo può assumere di standard sono:

1 Normale (su stampante predefinita)

2 eMail*

3 Pdf*

4 Fax*

* Necessaria implementazione personalizzata su script *Ambiente.vbs*.

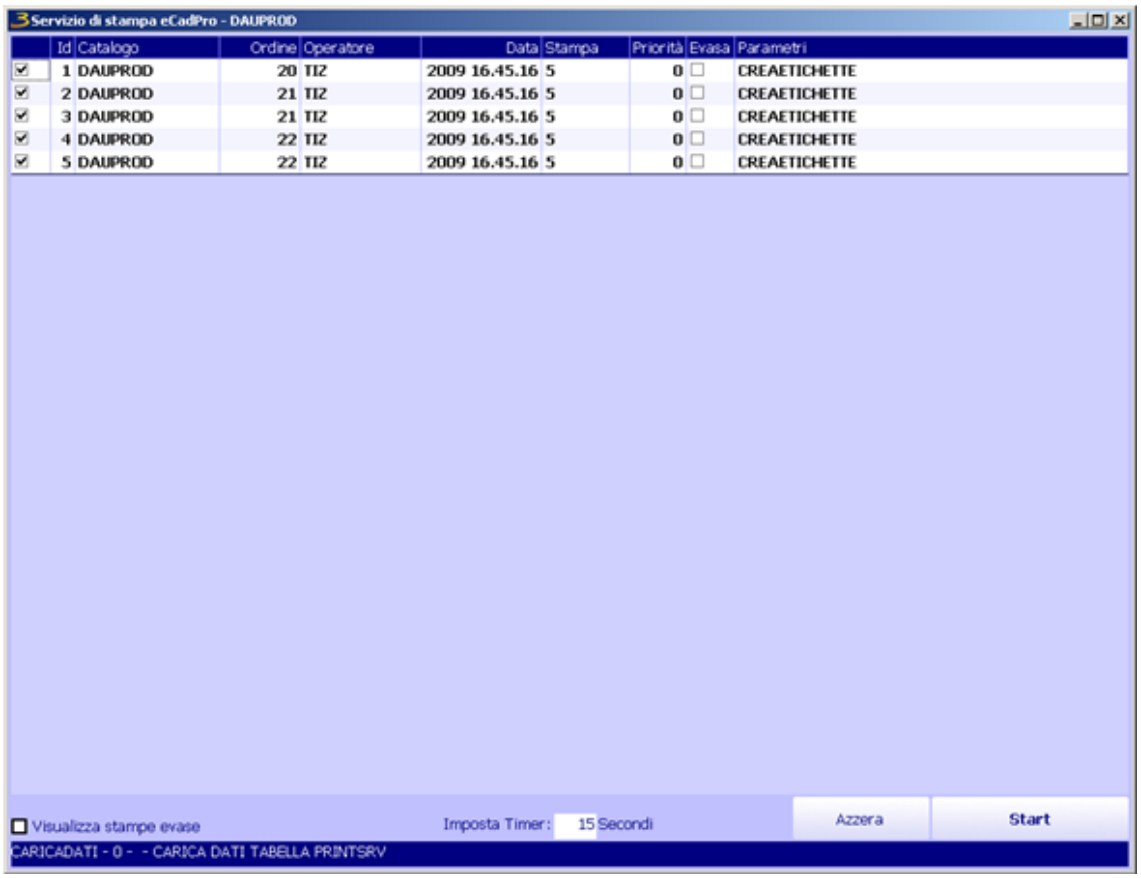

Servizio di stampa eCadPro - DAUPROD
Id Catalogo Ordine Operatore Data Stampa Priorità Evasa Parametri
1 DAUPROD 20 TIZ 2009 16.45.16 5 0 CREAETICHETTE
2 DAUPROD 21 TIZ 2009 16.45.16 5 0 CREAETICHETTE
3 DAUPROD 21 TIZ 2009 16.45.16 5 0 CREAETICHETTE
4 DAUPROD 22 TIZ 2009 16.45.16 5 0 CREAETICHETTE
5 DAUPROD 22 TIZ 2009 16.45.16 5 0 CREAETICHETTE
Visualizza stampe evase
Imposta Timer: 15 Secondi
Azzera
Start
CARICADATI - 0 - - CARICA DATI TABELLA PRINTSRV

Figura 10.2: Visualizzazione degli ordini da stampare.

Il campo Priorità definisce la priorità con cui l'ordine deve essere stampato, i valori numerici più bassi hanno maggiore priorità.

Il campo Evasa indica se un ordine è stato processato correttamente.

Il campo Parametri può contenere altre istruzioni che possono essere poi lette ed elaborate sempre all'interno dei file di script. Nel funzionamento standard, il primo parametro che viene indicato in questo campo è il modulo di stampa con cui deve essere stampato l'ordine. Attualmente sono gestiti anche altri comandi:

- CreaEtichette: esegue la stampa etichette;
- VBS: richiama la funzione `PrtSrvEsegui` all'interno del file *Ambiente.vbs*;

- CreaJpgOrdine: genera disegno ordine in formato JPG.

Esempio 2. Creazione del disegno ordine in formato JPG.

```
function CreaJpgOrdine(id, objPrintSrv)
Dim nFile,res
nFile = "c:\" & Ambiente.xPercorso & "_" & Xamb.info.
    NumeroStr & "_1D.JPG"
res=ambiente.CreaFileArg (cStr(nFile),"/V1 /SL1 /CO /Z0.85 "
    ,1024,768)
nFile = "c:\" & Ambiente.xPercorso & "_" & Xamb.info.
    NumeroStr & "_2D.JPG"
res=ambiente.CreaFileArg (cStr(nFile),"/V2 /SL1 /CO /Z0.85 "
    ,1280,1024)
nFile = "c:\" & Ambiente.xPercorso & "_" & Xamb.info.
    NumeroStr & "_3D.JPG"
res=ambiente.CreaFileArg (cStr(nFile),"/V3 /SL1 /CO /Z0.85 "
    ,1024,768)
nFile = "c:\" & Ambiente.xPercorso & "_" & Xamb.info.
    NumeroStr & "_4D.JPG"
res=ambiente.CreaFileArg (cStr(nFile),"/V4 /SL1 /CO /Z0.85 "
    ,800,600)
end function
```

Altri campi della tabella che non sono visualizzati nella finestra sono il campo Output che definisce con quale stampante deve essere stampato l'ordine e il campo Escludi che indica quale ordine deve non deve essere processato dal printSrv.

10.3 Download ordini

Attraverso la procedura di download, è possibile scaricare in automatico gli ordini dei propri clienti inviati alla sede tramite ***eCadLite*** ; una volta che gli ordini sono stati scaricati viene chiamata in automatico la procedura di stampa di questi ordini secondo quanto già visto nella documentazione.

La tabella a cui fa riferimento questa procedura è la xtOrdineConf che è situata nel database eCadMaster on-line.

Requisito fondamentale affinché questa procedura possa funzionare sono l'attivazione della postazione adibita al download tramite il supervisore (SV) e l'impostazione della voce connettiInternet uguale a 1 nell' *ecadpro.ini* in `c:\windows`, gruppo [INFOGEN]. In caso contrario verrà notificato che la postazione non è abilitata al download e la procedura verrà interrotta.

La schermata di download è diversa rispetto a quella di stampa in quanto vengono visualizzate informazioni riguardanti ad esempio il numero ordine scaricato, il mittente, lo stato ecc., ogni ordine scaricato correttamente viene marchiato come scaricato e sarà quindi ignorato al successivo download.

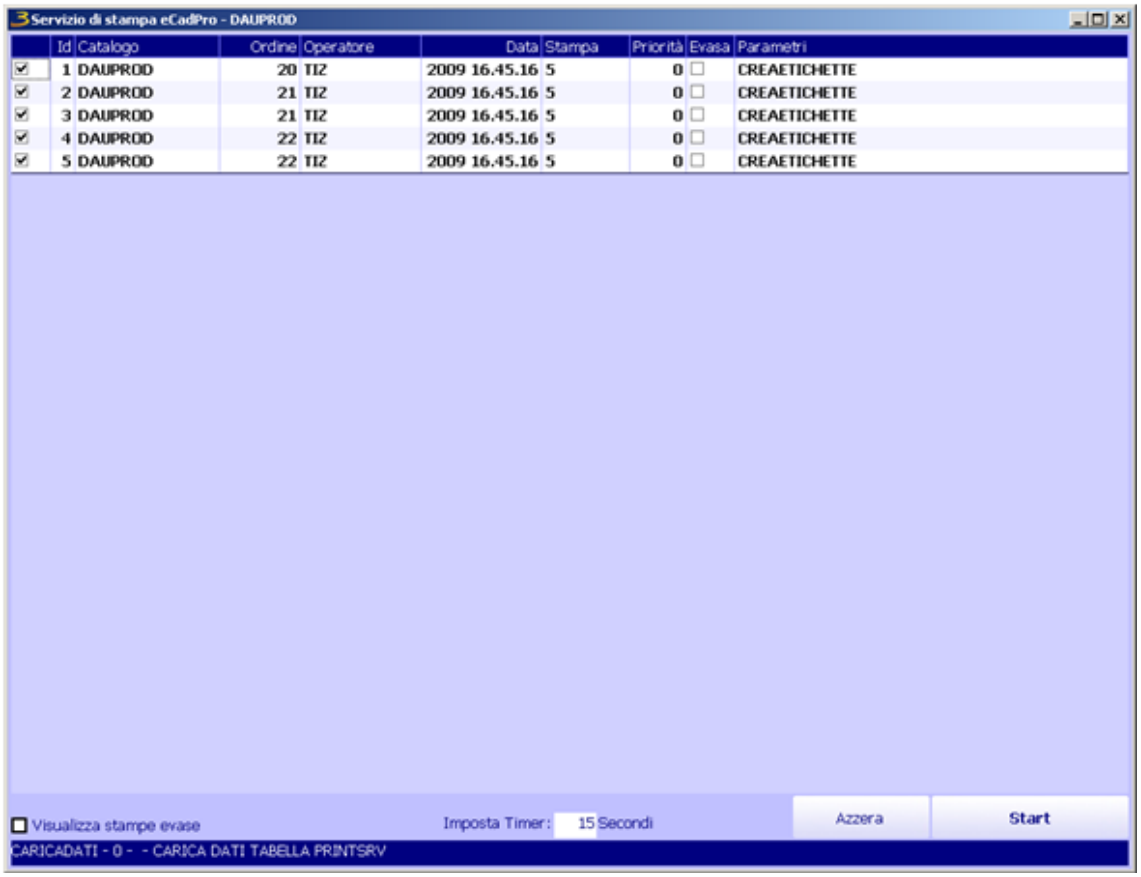

Servizio di stampa eCadPro - DAUPROD
Id Catalogo Ordine Operatore Data Stampa Priorità Evasa Parametri
1 DAUPROD 20 TIZ 2009 16.45.16 5 0 CREAETICHETTE
2 DAUPROD 21 TIZ 2009 16.45.16 5 0 CREAETICHETTE
3 DAUPROD 21 TIZ 2009 16.45.16 5 0 CREAETICHETTE
4 DAUPROD 22 TIZ 2009 16.45.16 5 0 CREAETICHETTE
5 DAUPROD 22 TIZ 2009 16.45.16 5 0 CREAETICHETTE
Visualizza stampe evase
Imposta Timer: 15 Secondi
Azzera
Start
CARICADATI - 0 - - CARICA DATI TABELLA PRINTSRV

Figura 10.3: Visualizzazione download ordini.

Gli ordini verranno scaricati in base a quanto impostato nella voce OrdiniInternet, gruppo [INFOGEN] dell'*ecadpro.ini* di sistema.

10.4 EvoStarter

E' un programma che permette di ovviare ad eventuali problemi di overload della memoria del PC sulle postazioni adibite al download e stampa degli ordini.
Il suo funzionamento consiste nell'avviare un ciclo in cui viene riavviato in automatico il Print Server ogni volta che questo si chiude. Per il corretto funzionamento di questa procedura occorre configurare uno script chiamato InfoPrintServer nel file *Ambiente.vbs*.

Avvio programma e parametri:

```
C:\evolution\ecadpro\ eCadProStarter.exe
C:\evolution\ecadpro\eCadPro.exe /START printsrvpro.crea
```

10.5 Elenco chiamate script utilizzate dal Print Server

InfoPrintServer (maxCiclo, autoExit, xTimer, maxDownload)

- maxCiclo: imposta il numero massimo di righe visualizzate ed elaborate
- autoExit: definisce dopo quante stampe il programma si chiude in automatico
- xTime: reimposta il timer del print server
- maxDownloaddefinisce: il numero massimo di download per ciclo elaborazione
- ggScad: definisce l'intervallo (in giorni) per la ricerca delle attivazioni in scadenza

PrtSrvTimerInizio (controlli)

Funzione per eseguire dei controlli iniziali. Il parametro comprende tutti i controlli form.

PrtSrvPreparaEsegui (id, controlli, ignoraComando)

Funzione usata per eseguire controlli sull'ordine che sta per essere elaborato. Se ignora comando viene impostato diverso uguale a 1 , l'ordine viene impostato direttamente come evaso e non viene elaborato.

PrtSrvEsegui (id, controlli)

Funzione che viene richiamata solo nel caso in cui il parametro di elaborazione è VBS.

CheckPdfStampato (nres)

Utilizzata per eseguire verifiche sugli ordini stampati. Se nRes viene impostato diverso da 1, l'ordine non viene evaso e il programma tenterà di stamparlo nuovamente al successivo ciclo di elaborazione.

PrtSrvFineEsegui (id, controlli)

Funzione che viene richiamata a fine elaborazione

onMailScadLic(emailDest, emailAz, emailAge, dataScad, lngUtente, resEmail)

Funzione che viene richiamata quando viene passato il parametro /LIC alla linea di comando. Serve per visualizzare ed eventualmente avvisare tramite e-mail i clienti che hanno la licenze di attivazione ***eCadLite*** in scadenza. Di default la ricerca viene eseguita sulle licenze che hanno scadenza entro 2 giorni dalla data odierna, in alternativa è possibile cambiare questo parametro tramite la funzione `InfoPrintServer`. All'interno della sub onMailScadLic è necessario impostare e personalizzare uno script per l'invio delle e-mail.

A Glossario

Glossario

3DS Il file contenete un oggetto tridimensionale. 46, 55, 66--68, 178, 194

ASCII American Standard Code for Information Interchange: gruppo predefinito di 128 caratteri (base character set) o di 256 (extended) e corrispondente standard di rappresentazione binaria. 52, 84, 93, 97, 102, 161, 196

Backup Procedimento di ricopiatura dei file per evitare la perdita dei dati. 19, 21, 22, 38

BCP Il file di backup del catalogo. 18, 19, 21

Bitmap Formato per l'archiviazione e l'interscambio di immagini in cui ai pixel dello schermo vengono fatti corrispondere gruppi di bit che ne descrivono le caratteristiche di ognuno. 55, 193

CAD Indica il settore dell'informatica volto all'utilizzo di tecnologie software e in particolare della computer grafica per supportare l'attività di progettazione (design) di manufatti sia virtuale che reali. 52, 66

Client Una componente che accede a servizi e risorse offerti da un'altra componente detta server. 10, 13, 15, 22

DLL Dynamic Link Library. moduli applicativi che contengono routine e/o risorse che vengono linkate all'applicazione chiamante solo al momento dell'esecuzione. Se le applicazioni richiamano routine comuni dalle librerie del sys e le link-editano nel loro codice oggetto (early binding) o le esigono già caricate in memoria, ne deriva uno spreco di spazio nelle librerie applicative o di memoria RAM. Con le DLL, invece, le routine vengono richiamate solo all'atto dell'uso effettivo (late binding), utilizzando eventualmente la copia già in memoria se altre applicazioni l'hanno precedentemente caricata. 6--8, 82, 83

Dump L'operazione di riversamento (in effetti la traduzione letterale del termine è discarica) di un qualcosa non meglio definito che può essere il contenuto della memoria. 20

DWG DraWinG exchange format. Estensione dei file di AutoCAD. 67

DXF Drawing Interchange Format, o Drawing Exchange Format è un formato per i file di tipo CAD, sviluppato da Autodesk come soluzione per scambiare dati tra il programma AutoCAD e altri programmi. 66

Log Per log si intende un registro nel quale vengono salvate delle informazioni utili per capire gli errori generati dall'applicazione nel corso del suo funzionamento. 22

PopUp Apertura di una finestra che si apre sopra la finestra precedente, imponendosi alla lettura. 42

Query Interrogazione. Nei database indica il criterio in base al quale si effettua la ricerca (select query) o la modifica (action query) di un particolare record o insieme di record. 115, 119

Server Per server si intende una macchina remota solitamente dalle capacità di calcolo molto superiori a quelle di una macchina domestica che offre servizi e risorse limitati. Nel modello gerarchico Client/Server il Server e' il sistema che fornisce un servizio richiesto da un altro sistema Client. 13, 15, 18, 82

Setup Comando per installare un programma o una serie di programmi, a volte durante installazione è necessario indicare delle preferenze o una configurazione. 7, 10, 38, 52, 82

SQL (Structured Query Language)Linguaggio di interrogazione per le basi di dati relazionali. 7, 20, 164, 165

Stored procedures E' un programma scritto in SQL od in altri linguaggi, mantenuto nel database stesso. Servono per evitare di riscrivere centinaia di volte la stessa query. 6, 10, 19, 21, 24

Texture Si tratta di immagini che sono uguali tra di loro, costruite in modo tale che quando si affiancano formano un rivestimento uniforme, senza che si possa più stabilire dove finisce una ed inizia la seguente, per tutti i lati di ogni immagine. 46, 47, 49, 50, 193, 195

Upgrade Modifica dei dati o di programmi contenuti in un computer. Le nuove versioni dei programmi, solitamente, presentano delle nuove funzionalità e la correzione degli errori presenti nelle precedenti. 11

Windows Abbreviazione di Microsoft Windows, è una famiglia di sistemi operativi commerciali della Microsoft Corporation. 47, 91, 105, 168